CDMetaPOP

USER MANUAL

2015

Version: 1.0

Last Updated: 2015.12.14(ell)

Authors:

Erin L. Landguth[1], Andrew Bearlin[2], Casey Day[3], Jason Dunham[4]

1 – University of Montana, Division of Biological Sciences, Missoula, MT, 59812, USA. erin.landguth@mso.umt.edu.

2 – Seattle City Light, Environmental Affairs Division, Seattle, WA, USA. andrew.bearlin@seattle.gov

3 – Purdue University, Department of Forestry and Natural Resources, West Lafayette, IN 47909, USA. day29@purdue.edu.

4 – USGS, Corvalis, Oregon, USA.

Table of Contents

1. Introduction 4

1.1 Changes from CDPOP/CDFISH 4

1.1.1. Major changes 4

1.1.2. Major extensions 5

1.2 What can CDMetaPOP do 6

1.3 How does CDMetaPOP work 7

1.3.1 Main Processes 7

1.3.2 Module details 19

2 Getting Started 32

2.1 Dependencies 32

2.1.1 Baseline requirements 32

2.1.2 Python on non-windows platforms 32

2.1.3 Python on windows 32

2.1.4 Obtaining NumPy and SciPy 32

2.2 Installation 33

2.2.1 Installing Python, NumPy, and SciPy 33

2.2.2 Installing CDMetaPOP 33

2.2.3 Description of CDMetaPOP files 34

2.3 Example run 34

2.3.1 Command line run 34

3 Input 37

3.1 Patch level controls – PatchVars.csv file 37

3.2 Class level controls – ClassVars.csv file 45

3.3 Run parameters and output – PopVars.csv file 50

3.4 CDClimate and movement surfaces with functions 54

3.5 Reproduction options 59

3.6 Offspring/litter/egg options 61

3.7 Genetic options 63

3.8 Spatial selection options 65

3.9 Infection options 67

3.10 Growth options (see Section 1.3.1.6) 67

4 Output 69
4.1 Individual files 69
4.1.1 ind.csv or indSample.csv 69
4.1.2 Genetics format option 71
4.1.3 Genetic distance files 71
4.2 Summary measures 71
4.2.1 Time versus class 71
4.2.2 Class by given time 72
4.2.3 Time versus patch 72
4.2.4 Patch by given time 75
5 Special Cases 75
5.1 Non-overlapping generations 75
5.2 Wright-Fisher assumptions 75
5.3 Deterministic and stochastic variability 77
5.3.1 Deterministic example runs 80
6 General Issues 91
6.1 How to obtain CDMetaPOP 91
6.2 Debugging and troubleshooting 91
6.3 How to cite CDMetaPOP 91
6.4 Disclaimer 92
7 References 92
8 Acknowledgements 96

1. Introduction

The goal of this user manual is to explain the technical aspects of the current release of the CDMetaPOP program. CDMetaPOP is a major extension and branch of the programs CDFISH (Landguth et al. 2012a) and CDPOP (Landguth and Cushman 2010; Landguth et al. 2012b). CDMetaPOP models genetic exchange for individuals through given resistance surfaces and *n – (x, y)* located patches as functions of individual-based movement through mating, migration and dispersal, vital dynamics (birth and death), density-dependent mortality, selection, and mutation. CDMetaPOP was developed for use in aquatic systems, but can be adapted to consider terrestrial systems as well.

Throughout this user manual a number of example models will be discussed. During development of the CDMetaPOP program, the primary species of interest were Westslope Cutthroat Trout (*Oncorhynchus clarkii*; WCT), Bull Trout (*Salvelinus confluentus*; BT), Lahontan Cutthroat Trout (*Oncorhynchus clarkii henshawi*; LCT), and Eastern Brook Trout (*Salvelinus fontinalis*; EBT). While the primary development was geared for riverine examples, we have also used the program with Whitebark Pine (*Pinus albauculis*) and Mountain Pine Beetle (*Dendroctonus ponderosae*), and thus, can be used for terrestrial systems as well.

1.1 Changes from CDPOP/CDFISH

We created CDMetaPOP in order to simulate 100s of thousands of individuals for more complex biological processes that are not included in CDPOP, CDFISH, or any other program to our knowledge. Therefore, major updates, extensions and additions to modules and functions within CDPOP/CDFISH were added that created a branch in these programs with modifications necessary for significant performance time speed ups. We list below the new functionalities for this program.

1.1.1. Major changes

- Metapopulation framework:
  - 100s of thousands of individuals can be tracked using a metapopulation framework where individuals are located in 'patches' on the landscape.
  - Patches can be designed to have a single individual (i.e., operating truly at the individual-based level) or a set carrying capacity with multiple individuals (i.e., operating at the deme-based level).
- Class structure: Age or size classes initialized by user and can control various processes in model, including
  - Maturation rate
  - Growth rate

  - Migration and straying probabilities
  - Fecundity
  - Capture probability
- Density dependence: Individuals are allocated space in 'patches' based on size and density dependent carrying capacity using a 'packing algorithm'.
- Movement rules:
  - Individuals move between 'patches', for mating, migration (to and from mating grounds), and local dispersal (straying).
  - Migration occurs at two separate time steps: 1) movement out of natal patches to an overwintering/foraging/rearing location and 2) movement back to natal patches from the overwintering/foraging/rearing location.
  - Individuals can fail to return back to natal patches and then can be considered a mortality event or can stray.
  - Straying can be modeled as a separate process during movement back to natal populations.
- Sex-based processes: processes are split up for sexes, including:
  - Maturation rate
  - Growth rate
  - Movement distances and probabilities
- Growth: Individuals grow based on von Bertalanffy equations (Bertalanffy 1983) and depending on where an individual settles during migration stages, as a function of temperature or other environmental surfaces.
- Temporal systematic, stochastic, and demographic variability: processes and parameters can vary through time and linked spatially to environmental or climate variables at the patch level (see Table 2):
  - Carrying capacity
  - Spatial selection surfaces
  - Environmental movement surfaces and rules
  - Species specific movement strategies
  - Harvesting and survival
  - Migration and straying probabilities
  - Temperature surfaces controlling growth rates and growing days.

1.1.2. Major extensions

- Natural selection: Spatial selection operates at different time steps (i.e., summer and winter) as migrating individuals settle into foraging and/or mating grounds. Selection can operate at the genotype-environmental level or through genotype-maturation and/or genotype-growth timing.
- All movement (mating, migration and straying) can be modeled separately, spatially, and temporally as functions of landscape/riverscape surfaces.

- Vertical transmission (infected parents to offspring) is included with giving probability and tracking of infected individuals.
- mtDNA option included.
- Output genotype option in multiple formats: GENALEX (Peakall and Smouse 2006, 2012), STRUCTURE (Pritchard et al. 2000), GENEPOP (Raymond and Rousset 1995; Rousset 2008), or a general format.
- Format from CDMetaPOP can be read into the newest version of PopGraph (Dyer 2014) in R (e.g., R Development Core Team 2012).
- Additional mutation models added: step-wise forward and backward.
- Burn-in procedure options for both population dyanmics, neutral genetics, and adaptive genetics.

1.2 What can CDMetaPOP do

Our goal for this program is to provide the ability to model more complex demographic and genetic ('demogenetic'; e.g., Labonne et al. 2008; Frank et al. 2011) processes for systems with hundreds of thousands of individuals. CDMetaPOP is a spatially-explicit, metapopulation-based program in which individuals move between patches. The program simulates the spatial patterns in individual-based genetic data as functions of individual-, patch-, and population-based demographics and movement as a function of the underlying riverscape/landscape structure. CDMetaPOP's representation of the spatial environment, population demography, and genetic processes provides a powerful framework to investigate the impact of ecological factors on the genetic and demographic structure of populations. This approach has already advanced knowledge of the patterns of genetic variation in spatially-explicit contexts. Example simulations have included:

- Quantifying the time to detect barriers (Landguth et al. 2010).
- Validating landscape genetics inferences and climate change applications (Castillo et al. 2014).
- Testing for the effects of population sample size and number of markers (Oyler McCance et al. 2012; Prunier et al. 2014).
- Assessing relative influence of adaptive versus neutral markers in detection of population genetic differentiation (Landguth and Balkenhol 2012).
- Comparing methods utility and approach in landscape genomics (Jones et al. 2014).

1.3 How does CDMetaPOP work

The program is written in Python 2.7 (v2.7.6) utilizing Numpy (Oliphant 2006) and Scipy (2001) packages and provided with installation instructions for most platforms, along with sample input files. CDMetaPOP is built on a driver-module, plug-in, docking architecture that allows for ease of future modular development. CDMetaPOP has been debugged as carefully as possible by testing all combinations of simulation options. Information for users, including user manual, FAQ, publications, ongoing research, developer involvement, and downloads can be found at http://cel.dbs.umt.edu/software/CDMetaPOP/.

1.3.1 Main Processes

1.3.1.1 Dynamic landscapes

Quantitative methods have a long history of application in the natural sciences. The use of models, ranging from simple heuristic approaches to highly theoretical and predictive tools, is widespread. However few simulation tools frame individual based numerical models within a spatially-explicit context. This approach is critical in natural resource management since many of the actions that managers have the ability to control are implicitly and inseparably connected to the landscapes that comprise the habitats of the subject biota. For example, simulations using this model can address how increasing temperature through growth and/or movement capability will affect demogenetic population responses, how insertion/removal of barrier features may potentially decrease/increase connectivity, or how habitat improvement, or harvesting actions through time, can effect demogenetic population viability. The ability to explicitly account for systematic changes in the landscape is critical not only to many aspects of applied management of populations, but also to the development of resource monitoring and evaluation plans that are sensitive to expected changes as a result of intervention. Consideration of dynamic landscape processes (e.g., the interaction between discrete environmental manipulations such as habitat improvements and Climate Change) is especially important in this context.

1.3.1.2 Natural selection

Past versions of CDFISH (Landguth et al. 2012) modeled three sources of genetic variation: gene flow, genetic drift, and mutation. These past versions assumed that different genotypes have an equal probability of surviving and passing on their alleles to future generations and thus, natural selection was not operating. CDMetaPOP now implements natural selection analogously to the adaptive or fitness landscape of allele frequencies (Wright 1932; Landguth et al. 2012). This new functionality enables extension of landscape genetic analyses to explicitly investigate the links between gene flow and selection in complex landscapes at an individual's level. Three different types of selection can be specified:

(1) Genotype-environment: The user specifies fitness landscape surfaces (values for each patch) for each genotype of a single diallelic locus or two diallelic loci that are under selection (one or two locus selection models). For example, three relative fitness surfaces must be specified for the three genotypes, AA, Aa, and aa, from the two alleles, A and a. Nine genotypes must be specified for the two-locus model. Selection is implemented through differential survival of dispersing individuals as a function of the relative fitness at the location on that surface where the dispersing individual settles. The program will continue all other processes the same with an additional step implementing selection during each of the migration processes. In addition, these spatial selection surfaces can vary through time and/or can be considered for only mature individuals.

(2) Genotype-maturation timing: The user specifies maturation curves (two parameters each for female and male probability of maturing) for each patch and for each genotype in the one-locus model. For example, AA could correspond to a slower maturation probability than aa and/or linked spatially to environmental effects.

(3) Genotype-growth timing: The user specifies growth curves (five parameters) for each patch and for each genotype in the one-locus model. For example, BB could correspond to a larger growth response curve than bb and/or linked spatially to environmental effects.

* Note that for options (2) and (3) that these process can be considered independent or linked. If independent, then locus A would control maturation timing and locus B would control growth. If linked, then locus A genotype determines the epigenetic response in locus B.

Simulations using the modules genotype-environmental associations can be used, for example, to warm high elevation streams and ask how quickly alleles for thermal tolerance "invade" these areas. Similarly, simulations can ask how alleles for earlier spawning and migration tied to changes in thermal and hydrologic regimes alter demographics in headwater streams. Simulations with the maturation timing liked to genotypes and spatially could be used to ask how commercial harvest could lead to fisheries-induced evolution, or linking warmer temperatures to quicker maturity.

1.3.1.3 Demographics

CDMetaPOP has the ability to model complex demographics through either age or size controls. The following describes how maturation, fecundity, mortality, and density dependence operates within the program.

1.3.1.3.1 Maturation

If age control is operating, then the probability of becoming a mature, reproducing individual for both a male and a female is given in the ClassVars.csv input file.

If size control is operating, then the probability of maturation for males and females is modeled using a logistic function of size (Downs et al. 1997).

p(mature) = exp(A + B * Length) / (1 + exp(A + B * Length)) (1)

where parameters A and B can be fit based on user length data. The default parameters are from Downs et al. (1997) length-maturity fit data.

In order to provide an over-riding control on maturation of very slowly growing animals, individuals will become mature at age 6 if they reach this age regardless of their size.

1.3.1.3.2 Fecundity

Mean number of offspring (eggs) per mature female can follow a Poisson deviate, normal deviate, constant draw, or random draw.

If age control is operating, then a mean and standard deviation for eggs per mature female is specified in the ClassVars.csv input file.

If size control is operating, then the mean egg number parameter is derived as a function of length with three functional forms available: 1) linear; (Downs et al. 1997), 2) exponential; (Bowerman 2013), and power (Schillings et al. 2011). The user may specify the parameters of the fit based on size data to fecundity relationships, or by using literature values. For example, in the WCT model, egg mean numbers to length data were fit with the following linear equation and parameters (Downs et al. 1997)

Number of Eggs = -790.7 + 6.2 * Length. (2)

In the BT model, an exponential fit was used (Bowerman 2013)

Number of Eggs = 9.5576 * exp (Length * 0.0181) (3)

In the LCT model, a power fit was used (Schill et al. 2010)

Number of Eggs = 0.0002 * Length ^ 2.5989. (4)

Egg laying frequency, iteroparity and semelparity (or skipped spawning) can also be considered for the population as a whole. Future versions will expand this function in more detail (skipped spawning as a function of length and environmental conditions; Rideout and Tomkiewicz 2014).

1.3.1.3.3 Mortality

Several options are available to regulate mortality. All mortalities are specified with a mean and standard deviation and are truncated for the minimum/maximum interval on the desired distribution. Note that future versions will consider a re-calculation on the unit interval to preserve the desired distribution (Todd and Ng 2001). Mortality variables can be specified to operate at the age class, size class, and/or patch level. These mortalities can be additive or multiplicative to simulate different rates of additional mortality operating on populations such as may be due to anthropogenic (e.g., harvest pressure) or environmental (e.g., flow or temperature; Ray 2007; Peacock and Dochtermann 2012) impacts.

At all 3 levels, two temporally independent, mortality values can be implemented: mortality out (applied to individuals of a given class when migrants are away from spawning grounds) and, mortality back (applied to individuals of a given class when back at spawning grounds). Patch level mortalities are compounded (exclusively mutual or independent) with the class specific mortalities such that exposure to any spatially-explicit mortalities can be specified for any or all of the age or size classes specified. See Figure 1 and 2 for a schematic on how the multiplicative (i.e., independent) and additive (i.e., exclusively mutual) mortalities are used, respectively.

Finally, spatially-explicit egg mortality (applied to egg class or eggs at birth location) is specified in the PopVars and/or PatchVars file. Note, that Age 0 class variables in the ClassVars file does not correspond to the egg class, but rather the emerged class that enters the population (e.g., fry or fingerlings).

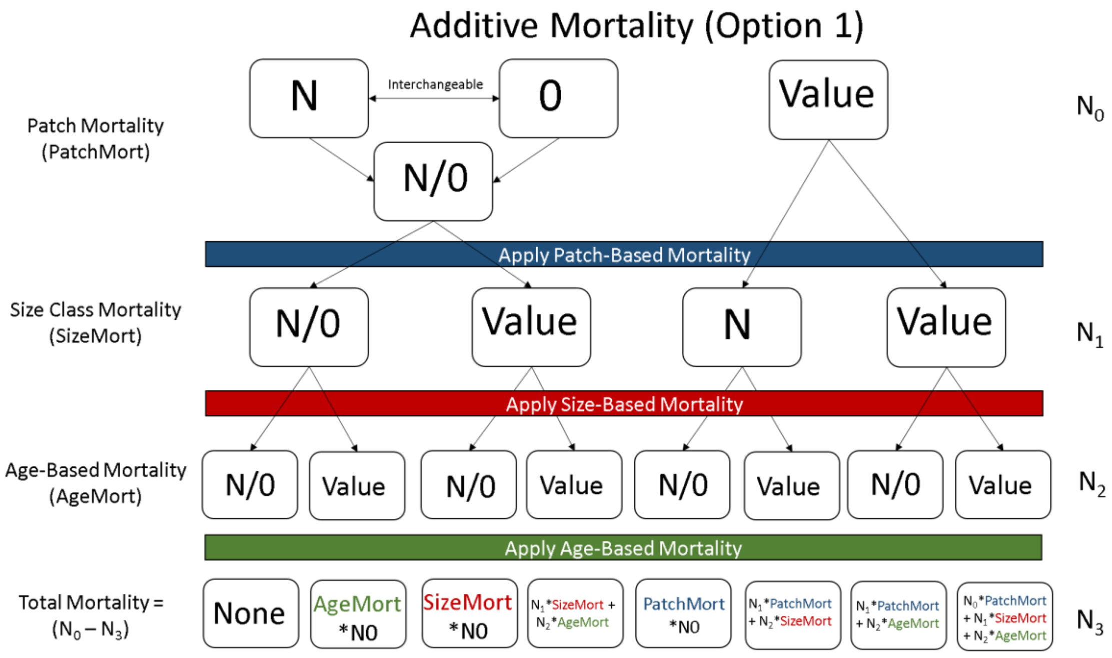

Additive Mortality (Option 1)
Patch Mortality (PatchMort)
N
Interchangeable
0
Value
N_0
N/0
Apply Patch-Based Mortality
Size Class Mortality (SizeMort)
N/0
Value
N
Value
N_1
Apply Size-Based Mortality
Age-Based Mortality (AgeMort)
N/0
Value
N/0
Value
N/0
Value
N/0
Value
N_2
Apply Age-Based Mortality
Total Mortality = ($N_0 - N_3$)
None
AgeMort *N0
SizeMort *N0
N_1*SizeMort + N_2*AgeMort
PatchMort *N0
N_1*PatchMort + N_2*SizeMort
N_1*PatchMort + N_2*AgeMort
N_0*PatchMort + N_1*SizeMort + N_2*AgeMort
N_3

Figure 1: Patch-, size-, or age-based additive mortality option (1).

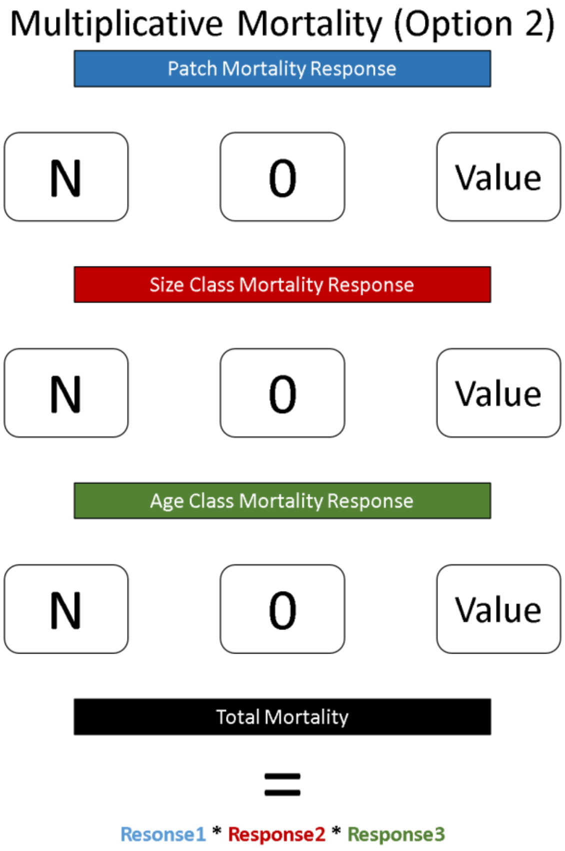

Multiplicative Mortality (Option 2)
Patch Mortality Response
N
0
Value
Size Class Mortality Response
N
0
Value
Age Class Mortality Response
N
0
Value
Total Mortality
=
Resonse1 * Response2 * Response3

Figure 2: Patch-, size-, or age-based multiplicative mortality option (2). Note that if 0 is the response for any category of mortality, all

other categories will be multiplied by 0 and result in no mortality at the relevant patch. If a response of 'N' is entered, then no mortality is used for that category. The combination of N * N * N will result in no mortality for the corresponding patch.

1.3.1.3.4 Density dependent mortality

In CDMetaPOP, individuals are able to move to new patches or reside where they are currently. Recruits are added into their natal patches. This means at each time step in the model, the number of individuals within each patch can potentially exceed resource capacity. The resulting level of mortality, reflecting competition for limited resources may be implemented via a simple "ceiling" function (i.e., population size will grow until patch carrying capacity (Kj) is reached using an exponential of logistic model), or via a density-dependent function with a hierarchical specification of patch- and class-structured carrying capacities (e.g., Ray and Hastings 1996).

The density-dependence is based on a modified Ricker model (Ricker 1954):

$N_{t+1} = N_t * \exp(R_0(1-N_t/K)$ (5)

with the ability to specify an ideal distribution of K among the population classes (age or length) and then by dynamically varying the "R_0" parameter of the Ricker equation, together with recursive reallocation of under-utilized capacity according to the relative abundances of each class competing within the patch. This simulates length or age based competition and partition of habitat patch resources, with the user able to specify the degree of overlap and relative influence of class capacity versus patch capacity.

When density-dependence is enabled, the number of individuals within each patch j is influenced at the overall patch level, as well as within each class i, by specifying patch carrying capacity (K_j), the distribution of carrying capacity across each class i in patch j ($K_{j,i}$), class abundance ($N_{j,i}$), and abundance of individuals within a patch (N_j). The following describes the steps taken to allocate space to individuals arriving at a given patch j. The following describes the steps taken to allocate space to individuals arriving at a given patch j:

Initially, we calculate the patch-level Ricker model moment (R_{0j}) for each patch given total individuals arrived (N_j) and carrying capacity for that patch (K_j):

$R_{0j} = \log(K_j / N_j) / (1 - N_j / K_j)$ (6)

with a special case if $K_j = N_j$, then R_0j = 1.

Then, given the patch level R_{0j}, the population of each class i within patch j ($N_{j,i}$) and the patch population (K_j), a class-specific, independent growth rate moment ($R_{0j,i}$) is calculated:

$$R_{0j,i} = \exp(R_{0j} * (1 - (N_{j,i} / K_j))). \quad (7)$$

This rate parameter is akin to the potential maximum growth rate of this class as a function of both overall patch level effects, and the intra-class competition effects (e.g., within cohort competition).

Next we describe the ideal distribution of K_j among classes as given by:

$$K_{j,i} = \exp(C * (i + 1)) \quad (8)$$

where C = - 0.6821, derived from length survey data in patch locations via the following method: Otolith data (38 otoliths; SCL unpublished data)were used to develop a von Bertalanffy age/length relationship which was then used to identify mean length for classes of interest identified in the model. Survey data from the local populations in the area of interest (954 Length measurements from 2012 survey information; West Fork Environmental, 2013) were then binned into these age/length classes and frequency analysis used to derive an observed class population structure. An exponential model was fitted to these frequency data expressed as proportion of overall abundance, with the assumption that gear selectivity results in under-representation of the smallest size classes. A negative lognormal likelihood was calculated for the predictive function and parameter estimation derived explaining expected proportion of K for each size class. Note, that this parameter, C, can be changed to fit user specific survey data.

Now, given $N_{j,i}$ $K_{j,i}$, and $R_{0j,i}$, the membership of each stage in each patch can be allocated a space. An interim variable ($N'_{j,i}$)is used to denote the temporary membership during the recursive process described below. Starting with the largest class i in patch j, and iteratively working to the smallest class, the proportion of individuals for the given class that can fit into j is given by:

$$N_{j,i}' = N_{j,i;t0} * \exp(R_{0j,i} * (1 - (N_{j,i;t0} / (K_j * K_{j,i})))) \quad (9)$$

If $N_{j,i;to} > N_{j,i}'$, then $N'_{j,i}$ individuals are randomly selected from $N_{j,i;t0}$ to survive to the next time step in that class i at patch j ($N_{j,i;t1}$) and the remainder are assumed to die. If $N_{j,i;t0} < N'_{j,i}$ then all of the $N_{j,i;t0}$ individuals survive to $N_{j,i;t1}$. The remaining carrying capacity ($N'_{j,i} - N_{j,i;t1}$) from that class is then made available to the remaining classes proportionate to the original distribution of K_j (equation 7).

$$K_{j,i}' = K_{j,i} - (N_{j,i;t1} * K_{j,i} / N'_{j,i}). \quad (10)$$

Finally, the new available habitat $K_{j,i;t1}$ is calculated for the remaining classes as

$$K_{j,i;t1} = \exp(-C * (i + 1)) + K_{j,i}' / \sum_{0}^{i-1} N_{j,i} \quad (11)$$

The recursive formulation of equation 10, beginning with the largest (or oldest) class, simulates a dominance hierarchy in allocation of space to different classes. This provides a density dependent mortality on each patch as a function of $N_{j,i}$, N_j, and K_j. Note that there can be rare situations in which the new calculated $N_j > K_j$.

The 'ceiling' option can be implemented by turning off the 'packing' option and then population size will grow exponentially until K_j in each patch is reached. Extra individuals will be discarded at random with no preference for class. In addition, K_j is allowed to vary at each time step with given values that can be function of a known variable, such as temperature, at each patch.

1.3.1.4 Patch versus class level controls

The user has the ability to choose parameters that operate at the patch, class, or both. Patch level controls can, for example, be used to define life history strategies (e.g., resident versus migratory individuals via patch controls on migration rates) or implement environmental patch factors (e.g., mortality linked to flow) or harvesting at spatial locations along the network (by adding a patch and/or class specific mortality). In order to turn off class level controls, set all probability parameters within the class file to 1 and mortality parameters to 0. The additional complexity of class level controls would mimic behavioral differences between classes. Furthermore, the class level controls and specific input file (e.g., ClassVars.csv) is specified in the patch level input file (e.g., PatchVars.csv). This allows for different class level controls to operate spatially. Also, there are a few additional population level controls that operate on all patches and classes equally (e.g., environmental movement functions and thresholds).

1.3.1.5 Age versus size control

The user has the ability to choose parameters that operate based on age or size. Some parameters, such as the constant mortality, can be controlled for both age and size. An example of this usage could be when a mortality is implemented for the oldest age class, another may be the case where harvesting is differentially applied to larger size classes.

1.3.1.6 Growth

Initialization of body sizes for individuals in the model is accomplished by specifying a mean and standard deviation of length in the ClassVars.csv input file. Size is allocated using a normal draw for each age class, i.

There are four growing options: (1) 'N' – turn off growth altogether, (2) 'known' – grow individuals at a known increment each year which is specified in the ClassVars input file with given age, (3) 'vonB' – grow individuals based on the von Bertalanffy equation (12) below, or (4) 'temperature' – grow individuals based on spatially-explicit temperature values and von Bertalanffy equations (12-15) below. Population growth rates are then specified in the PopVars.csv input file. Growth can occur at two times of the year, when they are back at their natal grounds or when they are away from their natal grounds.

1.3.1.6.1 Von Bertalanffy growth

The option 'vonB' will grow each individual based on the von Bertalanffy equation (Bertalanffy 1938),

$$L = Loo * (1 - \exp(- k * (i + 1-t_0))) \qquad (12)$$

where Loo and R0 are the theoretical maximum size and growth rate, respectively, i is the age class, and L is the new size.

1.3.1.6.2 Temperature and statistical fitting for growth

The option 'temperature' will use spatially-explicit temperature values at each patch location to drive growth for each individual, depending on where (i.e., patch) the individual is located, X_j. This option requires the user to specify three key elements: (1) von Bertalanffy parameters in Eqn. 12, (2) the temperature-growth response curve, and (3) the length of the intra-time unit (e.g., 'season') during which individuals are exposed to different temperatures at patches (i.e., grow days). In example models presented here, literature-derived temperature-growth curves and the von Bertalanffy model were fit to empirical data (e.g., field and hatchery) to obtain observed von Bertalanffy parameters at the observed temperatures and derive the values of X_max and X_CV that describe the temperature-growth curve approximated to a normal distribution. With optimal von Bertalanffy parameters, and temperature-growth relationship specified, an adjusted k (k') (based on patch temperature) is used to calculate the seasonal increment in length (L_inc):

k’ = k * norm(X_max,X_CV*X_max).pdf(X_j) /

norm(X_max,X_CV*X_max).pdf(X_max) (13)

and

L’ = Loo * (1 – exp(k’ * (i + 1 – t_0))) *

norm(X_max,X_CV*X_max).pdf(X_j) /

norm(X_max,X_CV*X_max).pdf(X_max). (14)

Then, at the given time step, t,

L_inc = L’ * exp(k’ * (i +1)) * (GD / 365) (15)

which gives the incremental growth given the environmental variable value, X_j, at patch j, adjusted for the grow days (GD) during that time period. Parameters X_max, X_CV, are the growth maximum and coefficient of variation that describe growth with respect to temperature. Figure 3 shows the resulting von Bertalanffy surface across a range of temperatures (2-20 degrees Celsius) for the WCT model.

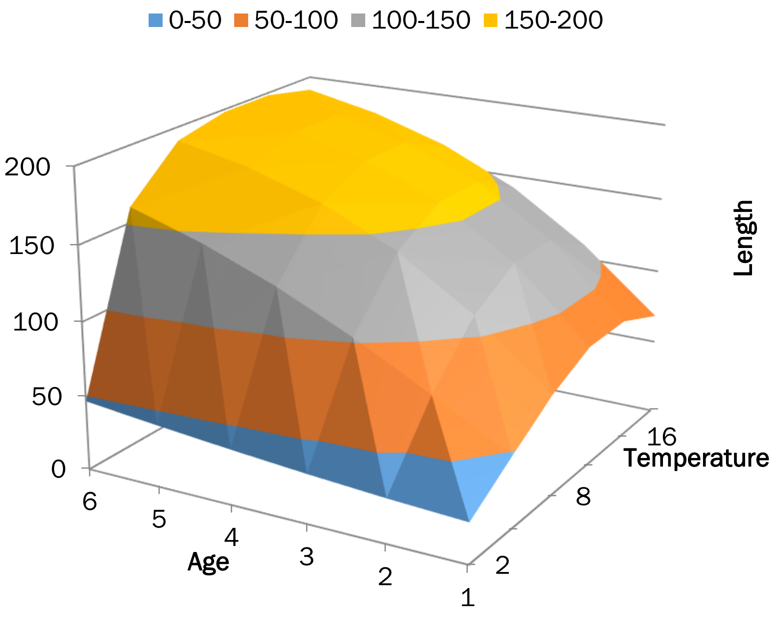


Figure 3: Age-length-temperature growth data (Creston National Fish Hatchery; Piper et al. 1982; unpublished SCL data; Bear 2005)

1.3.1.7 Individual movement

Landscape structure can govern movement of individuals between patches and this concept is incorporated through resistance (or permeability) surfaces whose values represent the stepwise cost from patch-to-patch of crossing each location (Spear et al. 2015). The model simulates movement as probabilistic functions of cumulative cost across these resistance surfaces (e.g., Landguth and Cushman 2010). Examples of types of cost distances from each patch to every other patch include least-cost path, Euclidean, or riverine distance. These distance matrices can be calculated from any program the user chooses (e.g., PATHMATRIX; Ray 2005, CIRCUITSCAPE; McRae 2006, UNICOR; Landguth et al. 2012c, 'gdistance'; Ettan 2015).

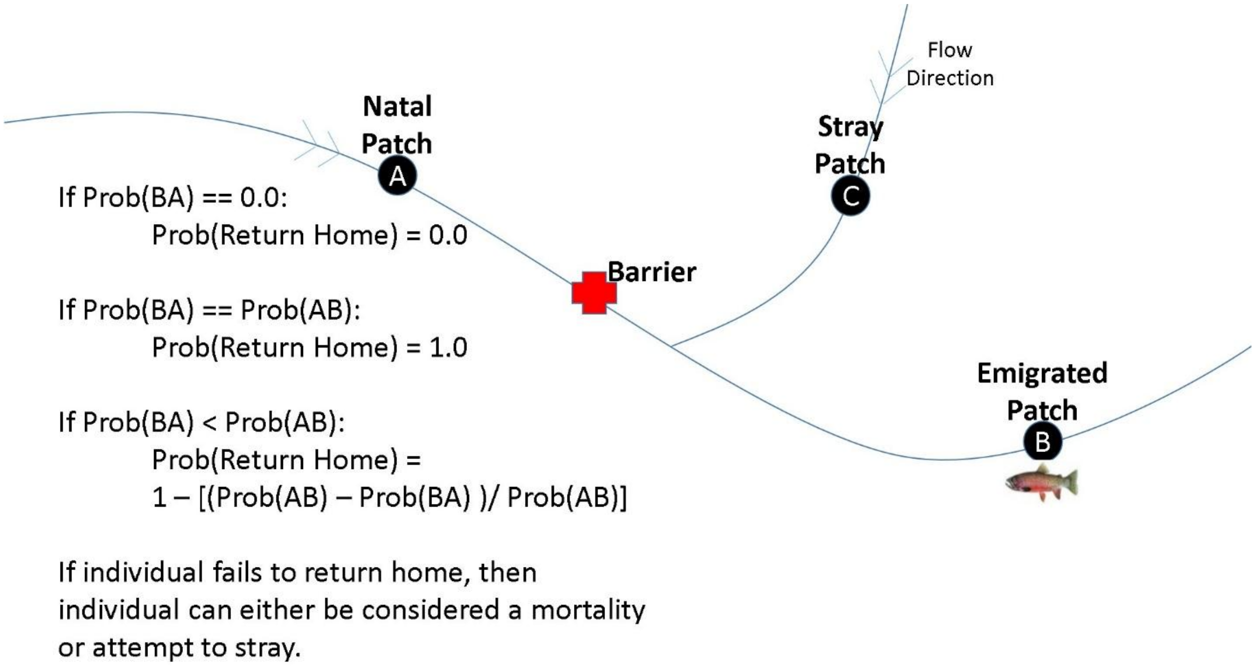
CDMetaPOP considers four effective distance matrices (mating, migration out, migration back, and straying) that are *j* X *j*, controlling where spatially an individual can move to as well as determining mate partner(s). There are 9 movement functions with threshold options that determine the probability of moving to a patch as a function of effective distance. The movement functions with thresholding can be used to determine the dispersal kernel for an individual from its natal patch and also help determine the most probable mate partner. The movement functions are linear, inverse square, negative exponential, Gaussian, nearest-neighbor, random mixing, within patch random mixing, and normalized distance. With the nearest-neighbor movement function, an individual moves to the available patch location nearest its initial location (or chooses a mate in a patch nearest to its own patch). Random mixing moves an individual to a patch location that is randomly chosen from all the patches on the landscape. Within patch random mixing would only move individuals or choose a mate within the same patch (most useful for mating options or restricting individuals to patches, e.g., ponds). In linear, inverse-square, negative exponential, and Gaussian movement functions, individuals move a distance from their initial location based on a draw from a probability distribution proportional to inverse square, negative exponential, or Gaussian, and inversely proportional to a linear. For example,

prob(move to patch j) =

D * 10^(-E * effective distance to patch j) (16)

is the negative exponential that returns a probability for movement to a given patch as a function of the effective distance to that patch with user specified parameters D and E (see Table 1 for a complete list). With normalized distance, individuals are placed based on a draw from the probability distribution inversely proportional to the rescaled maximum and minimum effective distance distribution. The user specifies the maximum dispersal distance (in effective distance units) an individual can travel on the landscape. The probability is one at no distance from the original location and goes to zero at the maximum dispersal distance. There is an additional option in which probability values from patch to patch can be considered or a user can convert effective distance values beforehand to probability of movement.

The movement functions and matrices can be different for each sex, (e.g., controlling for sex-biased dispersal). In addition, asymmetrical effective distance matrices can be given, for example, to test hypotheses in hilly terrain or one-way versus two-way barriers. The simulation program assumes these matrices are ordered by column, then row, so it is important to format asymmetrical matrices correctly (transpose if necessary). In other words, if patch 2 is uphill of patch 4, then the cost of patch 2 to patch 4 < the cost of patch 4 to patch 2. This is reflected in the columns of the cost matrix (i.e., matrixValue[column,row]) or matrixValue[2,4] < matrixValue[4,2]. Note that if following the 'gdistance' vignette example, then no transpose is necessary. See Figure 4 below for how probability of migration back to natal grounds is considered.



If Prob(BA) == 0.0:
    Prob(Return Home) = 0.0

If Prob(BA) == Prob(AB):
    Prob(Return Home) = 1.0

If Prob(BA) < Prob(AB):
    Prob(Return Home) =
    1 – [(Prob(AB) – Prob(BA))/ Prob(AB)]

If individual fails to return home, then individual can either be considered a mortality or attempt to stray.

Figure 4: Probability matrices (patch X patch that can be calculated based on effective distance matrices) are used to spatially place individuals in the landscape. Individuals can move based on 3 processes (probability matrices): migration out from natal grounds, migration back to natal grounds, and straying. Probability matrices can also be used to choose mate partners from other patches. An individual can fail to return home either because the Prob(return home) = 0 or the random chance it did not make it home due to Prob(BA) < Prob(AB). See flow diagram (Fig. 5) on movement order of each process.

1.3.1.8 Modeling capture probability

Detection or capture probabilities can be specified at the patch or class level. To only consider class level detection probabilities, enter a value of 1 for each patch. To only consider patch level detection probabilities, enter a value of 1 for each class. The program will produce 0/1 detection for each individual as well as recapture information. The option to capture individuals can occur when they are back at natal grounds, when they are away from natal grounds, or at both time periods. This flexibility allows complex capture dynamics to be simulated, e.g., class probabilities might be used to simulate size based gear selectivity bias, while patch probabilities might be used to represent spatially-explicit patterns of capture probability related to habitat complexity.

1.3.2 Module details

CDMetaPOP’s main architecture schematic and workflow is shown in Figure 5. Figures 5a and 5b show examples of timing events or seasonal processes that occur during one time loop of the model. Each module (box) is explained in further detail below with example parameterization for a Westslope Cutthroat trout model (Fig. 5a) for the Sullivan Basin in northeastern Washington.

1.3.2.1 PreProcess()

This module is for pre-processing of input files supplied by user. There are three input files to specify parameters and control for the population, patch, and demographics. In addition, users specify effective distance matrices that guide movement of individuals. Optional genetic files can be included to initialize population genetic structure. Table 1 explains each input file and parameters in more detail.

1) Population control: Example PopVars.csv files are given in the ../data/ folder. The user must specify the population wide input parameters through an input file. These are parameters that affect the population as a whole. As the model simulates stochastic processes, most applications will quantify mean and variability of population and genetic structure across many runs. Thus, a Monte Carlo option is provided for the user to choose the number of runs to simulate given a single set of input parameters. In addition, a user may also frequently wish to launch several runs with different parameter values simultaneously (i.e., sensitivity analysis). This

functionality is provided through batch capability.

2) Patch control: Example PatchVars.csv files are given in the ../data/ folder. The user must specify *j – (x, y)* located patches through an input file with information on carrying capacity (K) and initial starting numbers (N_0) for each patch, *j*. The program simulates a density dependent K within each patch as a function of classes (see 'packing algorithm' above). Patches will populate and fill based on the packing algorithm, and the ability of individuals to move to and out of a patch is a function of the riverscape or landscape resistance surface. $N_{0,j}$ individuals are assumed to occupy random locations within each patch. The genotypes for each individual can be initialized by randomly choosing from a file containing allele frequencies for each locus or random assignment (i.e., maximum allelic diversity). Patch level controls on migration and straying can be implemented to help define life history strategies. Patch level mortality on all classes or eggs can be given and linked to environmental/climate variables or harvesting data. Temperature, growing days, and environmental/climate values can be given for each patch that determine growth and/or differential mortality tied to 1- or 2-loci selection models. Most all of these parameters can change through time through the module CDCLIMATE and through stochastic temporal variability.

3) Demographic control: Example ClassVars.csv files are given in the ../data/ folder. The user must specify a file containing the number of classes and distribution. Then, the age, sex, maturity, and body size of each individual is initialized in each patch based on user specified information within this file. The sex of each initial individual is randomly assigned, but can be as an equal or unequal ratio. The body size (e.g., length) is drawn from a normal distribution with mean and standard deviation of body size supplied within this file. Probability of maturation for each sex may be implemented as a function of size, or specified for each age class. Fecundity can similarly vary based on size or be specified by class within this file. Age or size level controls on migration, straying and mortality may also be specified here to define class movement and survival.

4) Effective distance matrices: Example effective distance files are given in the ../data/cdmats/ folder. The effective distance (i.e., cost distance) matrices are n x n and govern how individuals move between patches. Four different matrices can be considered for the four different movement processes (mating, migration out of natal patch, migration back to natal patch, and local dispersal/straying). The same matrix can be used for all processes and the ability to turn off some of the

process is an option through the use of movement rules (e.g., in terrestrial system you may want to consider only migration out and/or local dispersal, but not migration back as with natal homing in some aquatic systems). In addition, at any year these matrices can be replaced to simulate changing resistance surfaces in the module CDCLIMATE. The effective distance matrices can be calculated with any connectivity algorithm (e.g., least-cost path in UNICOR (Landguth et al. 2012c), ArcGIS, or the package 'gdistance' in R (Ettan 2015), or circuit theory (e.g., McRae 2006)).

5) Genetic files: Example allele frequency files with format needed for CDMetaPOP are given in the ../data/genes/ folder. A user can specify the genetics to be randomly assigned for each individual or provide allele frequency distribution files that can be linked spatially to different patches. Files can be used for each patch or groups of patches.

1.3.2.2 GetMetrics()

Various metrics for tracking are tallied and recorded in this module See section 4 for the complete list.

1.3.2.3 First Update()

An initial time step file will be written to the output directory, giving information for each individual, including genotypes, sex, age, size, maturity, and infection status.

1.3.2.4 Begin time loop

After preprocessing of input data, a time loop begins. In Figure 2, examples are given for the corresponding modules/processes and timing events for spring (Fig. 5a) and fall (Fig. 5b) breeders (e.g., in aquatic systems, different seasonal spawners).

Example: For the WCT model, the beginning of the timing loop corresponded to the initial census data of June 1.

1.3.2.5 CDClimate()

This module controls systematic and temporal variability. For systematic variability, new population and patch variables, as well as environmental surfaces can be read in at this step. For temporal variability, updates around mean population or patch variables can occur here.

1.3.2.6 Mate()

Reproduction is either hermaphroditic or heterosexual. With hermaphroditic mating, there are no distinct sexes, but individuals mate with other individuals according to the movement function choice (described below), exchanging genes in Mendelian reproduction. In heterosexual reproduction, mated pair options include polygamous, polyandry, and monogamy, which are specified through replacement options for males and females. For example in polygamy, mated pairs are one male (male replacement is 'Y') to possibly many or no females with females only mating once (female replacement is 'N'), and the end of the mating process occurs when all females have mated. Frequency of egg laying or 'skipped spawning' (Rideout and Tomkiewicz 2011) may also be optionally specified to simulate iteroparous or semelparous life histories.

Probability of maturation can be specified for male and female age/size classes and updated at this time as a function of size with fitted parameters following Downs et al. 1997:

$$p(\text{mature}) = \exp(\text{intercept} + \text{slope} * \text{Length}) / (1 + \exp(\text{intercept} + \text{slope} * \text{Length})) \qquad (17)$$

There are eight movement functions that define how individuals choose a mate on the landscape as a function of effective distance. For example, with the nearest-neighbor movement function, an individual chooses a mate that is nearest its location. Random mixing chooses a mate that is randomly chosen from the total individuals in the population or patch. In the linear, inverse-square, negative exponential, and Gaussian movement functions, mate partners are chosen a distance from their initial location based on a draw from a probability distribution inversely proportional to a linear, square, negative exponential function, and Gaussian distribution. The random options do not use the effective distance matrix, and mate choice is an equal probability for all. In addition, there is a choice to not supply a functional form, but instead lets the rescaled cost values choose mate partners (the choice of mate partners will then be dependent on the distribution of the effective distance matrix). The user specifies the maximum dispersal distance (in effective/cost units) an individual can travel on the landscape to find a mate. The probability is one at no distance from the original location and goes to zero at the maximum dispersal distance. There is also the option of converting cost values to probability values beforehand and reading in the probability matrix from patch to patch, in case a functional form is not available in the previous 8 movement options.

Example: In the WCT model, mating was without replacement for females and with replacement for males and occurred randomly within a maximum patch distance of 2 km (option 4). In addition, there were known patches that were not suitable spawning locations (i.e., set based on gradient, temperature and known barrier locations) and initialized with $N_0 = 0$. Probability of maturation was based from Downs et al. (1997) size data.

1.3.2.7 Offspring()

Four options for determining the draw of mean litter size or egg numbers are available: (i) Each mated pair can have a number of offspring that is a bounded random draw based on a uniform probability distribution, (ii) a Poisson draw with specified mean, (iii) a constant number or (v) a normal draw with specified mean and standard deviation. Mean litter size or egg number can be considered for multiple paternity, i.e., if a mean litter size was 8 and a female had 2 mate partners, then the female could either have 16 total eggs or 8 eggs split between the fathers. A deterministic egg number can be specified by using a constant egg number.

Fecundity can be determined based on the mother's class (age or size, i.e., the draw mean with standard deviation). If size is specified, then one of three functional forms (exponential, power or linear) may be used to model size-egg relationships with parameters developed from empirical data.

Sex assignment can be random or proportional to a percentage of female offspring specified in the population control file (PopVars.csv).

Mendelian inheritance with *k*-allele mutation (rate chosen by the user) is used to generate the Age 0's genotype. Various mutational model options are available (see Table 1). A minimum of two loci and two alleles per loci can be used up to as many loci and alleles per loci within limitations of computational resources (recommended loci X alleles < 10^5). SNPs can be simulated using 2 alleles and X number of loci, but it is recommended to simulate with more alleles per loci and sample to a SNP equivalent (Landguth et al. 2012d).

Disease dynamics can be considered here as well, passed along from the mother based on a vertical transmission probability specified in the population control file (see CDINFECT module).

Mortality of eggs or survival to fingerlings/young of year is also applied at this module. A draw mean mortality with standard deviation can be specified at the population level (PopVars.csv). In addition, spatially-explicit patch mortality can be considered in the patch level file (PatchVars.csv) and could be derived based on relationships for survival and environmental data, such as flow, desiccation, or scouring.

The age class is still assumed to be 0 for the survived individuals after the module with size initialized from a normal draw with given mean and standard deviation (specified in the ClassVars.csv). However, this age class is not included in the population until the *DoImmigration()* module (or eggs to fingerlings are assumed to be resident individuals their first year). The fingerlings then enter the population as the adult fish arrive back to natal patches. These fingerlings are then considered in the packing algorithm in the *DoImmigration()* module. Future versions will consider a gestation period which would tell the model how long to keep the Age 0 class as residents in order to address questions related to emergence timing.

Example: In the WCT model and following Downs et al. 1997, fecundity was modelled as a function of size, X, using a linear model with slope (Egg_Mean_par1) and intercept (Egg_Mean_par1) parameters to estimate mean egg number (*E_mu*).

$$\begin{aligned} E\_mu &= Egg\_Mean\_par1 + Egg\_Mean\_par2 * X \\ &= -790.7 + 6.2 * fork\ length\ (mm) \end{aligned} \tag{18}$$

Example: In the WCT model, mean and standard deviation for egg survival (Creston National Fish Hatchery; Piper et al. 1982; unpublished SCL data; Bear 2005) were specified and a normal deviate was generated (mean probability of mortality 0.624 with standard deviation 0.1872), which was truncated at tail probabilities (i.e., values less than 0 became 0 and values greater than 100 became 100). Additional mortality was considered for temperature thresholds and flow regimes (e.g., scouring) for each natal patch.

1.3.2.8 Second Update()

A second update for individuals is conducted in which body size is incremented as specified (see *Growth* description for more details above). Age is also updated here and individuals in the last age class are grouped with the last age class (e.g., Age 5+). The new Age 0 recruits are not considered in this update. The option of capturing (sampling) individuals can occur here as well. An option to write a file to output directory giving information for each individual, including genotypes, sex, age, maturity, id, infection status, movement distance, and body size information (see Section 4 for more details on output files).

Example: As described above, in the WCT model, growth was modeled as a von Bertalanffy process where the growth rate parameters (R_0) was dependent on temperature and growing days in the currently occupied patch. Hatchery growth data (Creston National Fish Hatchery), empirical data (unpublished SCL data) and published values (Piper et al. 1982; Bear 2005) were used to derive a temperature response curve for *R_max* from which a growth increment was calculated based on temperature, grow days, age and the von Bertalanffy parameters. For a Spring-spawning species, such as WCT, the growth at this time step represents non-overwintering growth, thus approximately 7 months (60%) of the year in the NE WA state study area.

For the time period in which individuals were back at their natal grounds, we used 123 grow days (June 1 – October 1) and extracted mean temperature values across this time range. The temperature model is described further below.

Temperature Model: Data from 13 temperature sensors, distributed across the WCT model study area, with observations spanning the 2014 water year, were used to develop gridded maps of stream temperature. Principal components analysis (PCA) was used to identify the spatial and temporal variation in the Sullivan Basin stream temperature data. For each site, daily average stream temperature time series were first calculated at each site from the raw 4-hourly observations. Three principal components captured 99% of the total variation among the 13 sites. The dominant principal component score represents the mean temperature across all sites, and is well correlated with measured temperatures at the Boundary Dam. The principal component loadings represent the spatial variation among sites. Regression models for each of the first 3 principal components were developed using elevation and flow accumulation extracted from spatial maps at each site. These regression models were then predicted to 30 meter (98 foot) resolution raster grids of Elevation and flow accumulation along the Sullivan Basin river network. Then, following Holden et al. (2011) stream temperatures were reconstructed at each grid cell by combining the modeled loadings and observed principal component scores, resulting in a daily predicted stream temperature map for the basin.

1.3.2.9 Emigration()

This module controls movement out of a patch (or natal population) for each individual. Multiple controls are available to specify a wide range of potential mechanisms for emigration. A probability for each patch can be used with an age/size class probability of moving. These are assumed independent events. In order to use just patch control, set all class migration probabilities to 1. In order to use just class control, set all patch migration probabilities to 1. In addition, the 'set migration' parameter can be used to fix migration for an individual (i.e., once an individual has migrated they continue to be an obligate migrator until death). If obligate migration is specified care must be taken to use the appropriate probability of becoming a migrant at a given age/size since rates will be cumulative across age/size classes.

Given these compounding probabilities of movement, the effective distance matrix with movement function and thresholds for emigration is used to place individuals in other patches (see the description in choosing a mate, as the same rules apply for choosing a patch location).

Example: In the WCT model, migration out of natal patches was assumed to occur during late Summer and Fall. We assumed equal probability of migrating for every patch (turn off patch migration by setting values to 1.0), but used 0, 0.1, 0.2, 0.3, 0.4, 0.4, … for the size class migration rates. Migration probabilities are unknown for WCT and these values were assumed.

To place individuals in an overwintering patch, we used an isolation-by-riverscape resistance hypothesis that considered directional cost in the river network using elevation and barriers. We used the 'gdistance' package (Ettan 2015) in R to calculate the least cost path along this river network considering direction. The calculation for asymmetrical cost followed the gdistance vignette example for hiking in a hilly terrain, modified with the following equation for the 'slope-speed' relationship. If slope <= 0, then speed (i.e., permeability or conductance) for a fish was assumed to be 1.0. If slope > 0, then speed was assumed to be:

$$\text{speed} = 2 /(1 + \exp(0.3 * \text{slope})) \qquad (19)$$

1.3.2.10 After Emigration Mortality()

Additional constant mortality for each patch or age/size class can be applied at this stage for Age 1+. These constant mortality modules could be linked with environmental attributes (e.g., flow regimes) or harvesting giving additional mortality across space. A user has the option to apply mutually exclusive or independent mortality events here.

1.3.2.11 Third Update()

A module for a third update for individuals in which body size is incremented as previously described, with the exception that the complement of the "growing season" parameter defined for each patch is used to calculate the growth increment appropriate at this time and patch location. An option to write a file to output directory giving information for each individual, including genotypes, sex, age, maturity, id, infection status, movement distance, and body size information (see Section 4 for more details on output files). This optional output to file could be used to assess sampling at different times within a year. Note, age is not incremented here, as it was incremented in previous second DoUpdate(). The option of capturing (sampling) individuals can occur here.

Example: In the WCT model, this time period occurs over winter for a spring spawner (but note that this time period could be over summer for fall spawners). We used 242 grow days for the period Oct 2 – May 31 and extract the mean temperature values from the daily temperature model (described above). Capture detection is not considered during the overwintering period.

1.3.2.12 Immigration()

This module returns individuals back to original patch (i.e., home) or disperses locally (i.e., strays; see Fig. 4). Again, both patch and age/size class specific probabilities for straying are available to the user and assumed to be independent events. In order to use just patch control, set all class straying probabilities to 1.0. In order to use just class control, set all patch straying probabilities to 1.0. If a successful straying probability event occurs, then these individuals move to any possible patch that the stray effective distance matrix with stray movement functions and thresholds allow. If an individual is not chosen for straying, then it is assumed to migrate back to its natal population. However, the movement back to natal populations is also based upon a dispersal back effective distance matrix with movement function and thresholding, allowing asymmetrical cost/probabilities of movement to be considered. Here, directional effective distance matrices could prevent individuals that traveled too far from returning back (e.g., one-way barriers). If Prob(Emigrated Patch to Natal Patch) = 0.0, then an individual cannot make it back to its original patch location. If Prob(Emigrated Patch to Natal Patch) = Prob(Natal Patch to Emigrated Patch), then an individual is assumed to return back to its original patch. Considering asymmetrical probabilities and directionality, if Prob(Emigrated Patch to Natal Patch) < Prob(Natal Patch to Emigrated Patch), then a random draw is taken to decide if an individual will return back based on the relative difference or 1 – Prob(Natal Patch to Emigrated Patch) – Prob(Emigrated Patch to Natal Patch) / Prob(Natal Patch to Emigrated Patch). If an individual fails to return back, then an option for a mortality event or additional straying chance can occur.

By specifying matrices for both movement out (Emigration() process), and movement back (Immigration() process), simulating directional movement is possible. The movement back can also restrict individuals to stay within their emigrated patches, which allows this movement option to not be considered (e.g., for terrestrial systems and one way dispersal).

Once individuals stray and/or make it back to natal populations, then they are subject to the density dependent sorting and packing algorithm based on class structure. Age 0+ are considered in this packing algorithm stage.

Example: In the WCT model, straying and migration back occurs during spring and summer for spring spawners. We assumed equal probability of straying for every patch (turn off patch straying by setting values to 1.0), but used 0.05 straying probability for each size class. For individuals that were successful under 0.05 probability of straying, we placed them into a new patch randomly, but only to patches that were suitable for spawning.

For individuals that did not stray, they were assumed to migrate back to their natal patches. We considered the same isolation-by-riverscape resistance hypothesis for migration out that considered directional cost in the river network using elevation and barriers (described above).

The Age 0 class are assumed to emerge as fry into the population with their size initialized around 31 mm with a standard deviation of 3 mm (Creston National Fish Hatchery; Piper et al. 1982; unpublished SCL data; Bear 2005).

1.3.2.13 After Immigration Mortality()

A constant mortality for each patch or age/size class can be applied at this stage for Age 0+. The constant mortality module could be linked with environmental attributes (e.g., flow regimes) or harvesting, to simulate additional mortality at this time step. A user has the option to apply mutually exclusive or independent mortality events here.

Example: In the WCT model, we applied a mortality catch of 75% to the last age class (Age 10) to simulate appropriate senescence of populations in the simulations. No additional size specific mortality or patch specific mortality was considered.

1.3.2.14 GetMetrics()

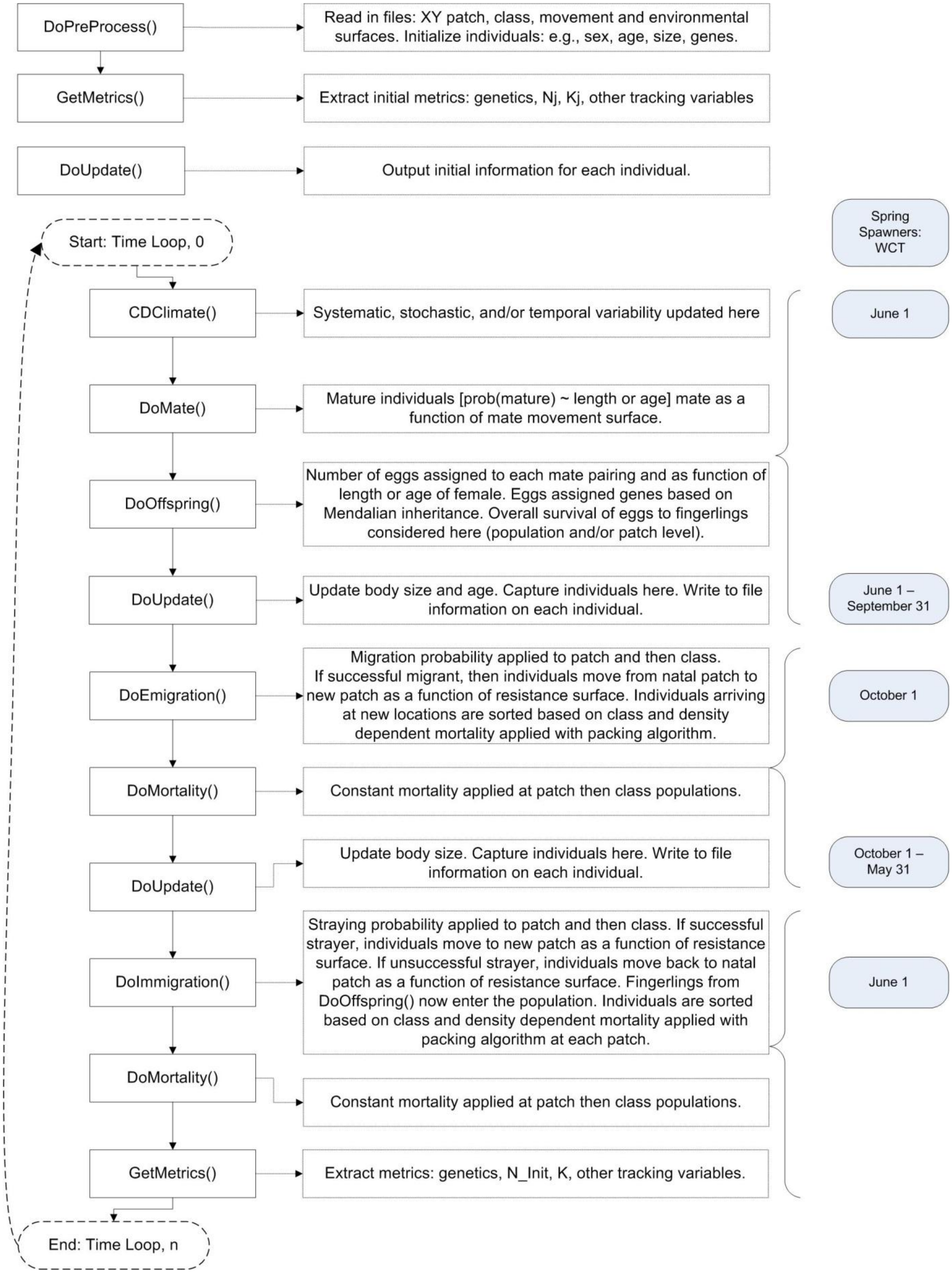
A second call to gather and track summary metrics within the time loop occurs here. A complete list is given in section 4.

1.3.2.15 End time loop

Repeat above processes until the end of simulation time (or generations) is reached.

1.3.2.16 PostProcess()

This module occurs once generations or years are completed. Tracking metrics through the GetMetrics() are summarized and written to summary files. Options are included due to space constraints for summarizing patch metrics at given times.



Read in files: XY patch, class, movement and environmental surfaces. Initialize individuals: e.g., sex, age, size, genes.

Extract initial metrics: genetics, Nj, Kj, other tracking variables

Output initial information for each individual.

Systematic, stochastic, and/or temporal variability updated here

Mature individuals [prob(mature) ~ length or age] mate as a function of mate movement surface.

Number of eggs assigned to each mate pairing and as function of length or age of female. Eggs assigned genes based on Mendalian inheritance. Overall survival of eggs to fingerlings considered here (population and/or patch level).

Update body size and age. Capture individuals here. Write to file information on each individual.

Migration probability applied to patch and then class. If successful migrant, then individuals move from natal patch to new patch as a function of resistance surface. Individuals arriving at new locations are sorted based on class and density dependent mortality applied with packing algorithm.

Constant mortality applied at patch then class populations.

Update body size. Capture individuals here. Write to file information on each individual.

Straying probability applied to patch and then class. If successful strayer, individuals move to new patch as a function of resistance surface. If unsuccessful strayer, individuals move back to natal patch as a function of resistance surface. Fingerlings from DoOffspring() now enter the population. Individuals are sorted based on class and density dependent mortality applied with packing algorithm at each patch.

Constant mortality applied at patch then class populations.

Extract metrics: genetics, N_Init, K, other tracking variables.

Figure 5a: CDMetaPOP main work flow schematic with timing for the example WCT system (spring spawning system).

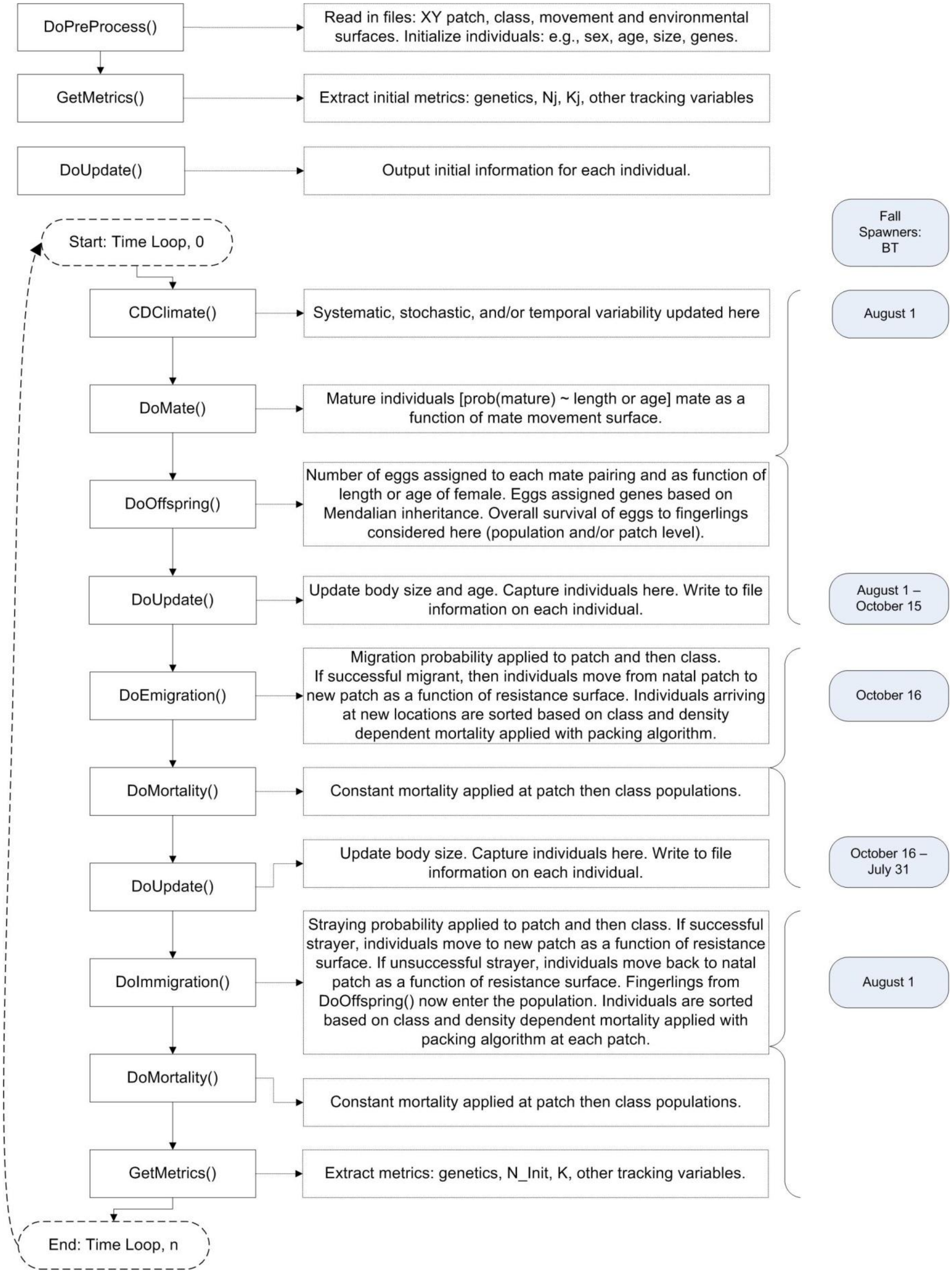

DoPreProcess()
Read in files: XY patch, class, movement and environmental surfaces. Initialize individuals: e.g., sex, age, size, genes.
GetMetrics()
Extract initial metrics: genetics, Nj, Kj, other tracking variables
DoUpdate()
Output initial information for each individual.
Start: Time Loop, 0
Fall Spawners: BT
CDClimate()
Systematic, stochastic, and/or temporal variability updated here
August 1
DoMate()
Mature individuals [prob(mature) ~ length or age] mate as a function of mate movement surface.
DoOffspring()
Number of eggs assigned to each mate pairing and as function of length or age of female. Eggs assigned genes based on Mendalian inheritance. Overall survival of eggs to fingerlings considered here (population and/or patch level).
DoUpdate()
Update body size and age. Capture individuals here. Write to file information on each individual.
August 1 – October 15
DoEmigration()
Migration probability applied to patch and then class. If successful migrant, then individuals move from natal patch to new patch as a function of resistance surface. Individuals arriving at new locations are sorted based on class and density dependent mortality applied with packing algorithm.
October 16
DoMortality()
Constant mortality applied at patch then class populations.
DoUpdate()
Update body size. Capture individuals here. Write to file information on each individual.
October 16 – July 31
DoImmigration()
Straying probability applied to patch and then class. If successful strayer, individuals move to new patch as a function of resistance surface. If unsuccessful strayer, individuals move back to natal patch as a function of resistance surface. Fingerlings from DoOffspring() now enter the population. Individuals are sorted based on class and density dependent mortality applied with packing algorithm at each patch.
August 1
DoMortality()
Constant mortality applied at patch then class populations.
GetMetrics()
Extract metrics: genetics, N_Init, K, other tracking variables.
End: Time Loop, n

Figure 5b: CDMetaPOP main work flow schematic with timing for a fall spawning system (e.g., BT or EBT).

2 Getting Started

2.1 Dependencies

2.1.1 Baseline requirements

CDMetaPOP requires the Python2.7.x interpreter, NumPy package, and SciPy package. Remember that Python modules usually require particular Python interpreters, so be sure the version ID for any external Python module or package (e.g., NumPy or others) matches the version of your Python interpreter (normally v2.7.x).

2.1.2 Python on non-windows platforms

Some common computer platforms come with Python installed. These include MAC OS X and most Linux distributions. To determine which Python a MAC or Linux workstation has installed, start a terminal console and enter "python." You'll see the version number on the top line (enter Control-D to exit). Replacing an older Python interpreter (pre v2.7) with a newer one (v.2.7.x) on a Linux or MAC OS X machine can be tricky, so ask a System Administrator for help if you're not sure which packages depend on the current Python installed.

2.1.3 Python on windows

Windows (8, 7, XP, 2000, Server) does not come with Python installed, so follow the instructions below to obtain and install Python on a computer running the Windows operating system. Get a windows installation of the base Python installation (current v.2.7.x) at:

http://www.python.org/download/releases/.

2.1.4 Obtaining NumPy and SciPy

A good suggestion for all-in-one installations is through the superpack Windows installer available from the SourceForge website: http://sourceforge.net/project/. Note that more complete information for NumPy is available at www.scipy.org, where the SciPy module is also presented. Another source is

http://www.enthought.com/products/epd.php for a free academic and educational usage in a single downloadable installer that has everything and then some (Numpy, Scipy, Matplotlib, and 70+ modules for python). We highly recommend this version for the easiest installs if you are new to Python.

2.2 Installation

2.2.1 Installing Python, NumPy, and SciPy

Make sure that Python and NumPy are installed, and available to you. You can test this by typing “python” at a command window. If python is available you will get the python prompt “>>>”. If it is not a recognized command, it means either that python is installed but is not in your command shell’s paths, or that python is not installed. In the first case ask an administrator to add it to your command paths. If your shell locates and loads python, type, “import numpy”. Similarly, type, “import scipy”. If python does not complain that there are no such modules, all is well.

The following instructions assume Python, NumPy, and SciPy are not yet available on your computer; if they are, skip to section 2.2.2.

> * First run the Python executable installer you’ve chosen (either from www.python.org, ActiveState, or EPD, accepting defaults for the installation directory. On Windows, this will typically place the executables and libraries in c:/Python2.7/bin and the “site-packages” package tree for user installed Python modules in C:/Python27/lib/site-packages. If you are installing it on a network on which you do not have administrative privileges, you may need to ask a system administrator to install python and the NumPy and SciPy packages in their default locations.
>
> * Next install NumPy and SciPy using the supplied executable (superpack) installer or visiting http://www.scipy.org/Download. This will install NumPy and SciPy in your Python ./site-packages directory. Note that if you choose EPD that you do not need to additionally install NumPy or SciPy.

2.2.2 Installing CDMetaPOP

Next, install the CDMetaPOP software itself by unpacking the zip archive supplied. Navigate to the directory on your PC where you wish to install CDMetaPOP, and unpack the supplied zip archive file using a free archive tool like 7Zip (7z.exe), Pkunzip, Unzip, or an equivalent. 7-Zip (7Z.exe) is highly recommended since it can handle all common formats on Windows, MAC OS X and Linux. On Windows, it is best to setup a project specific modeling subdirectory to perform your simulations outside of any folder that has spaces in its name (like “My Documents”). At this point you should be able to execute the supplied test inputs.

2.2.3 Description of CDMetaPOP files

Three directories will be installed in your directory. A description of each follows:

1. src – CDMetaPOP source code

2. doc – documents

    - README.txt – a quick how to run CDMetaPOP instructions
    - CDMetaPOP_user_manual.pdf – this file
    - CDMetaPOP_history.txt – Notes on history and version changes.

3. data – Example input files
    - cdmats/
        - Patch7_Cdmatrix.csv – example effective distance matrix file for 7 patches calculated using UNICOR (Landguth et al. 2012) for riverine distance.
    - genes/
        - allelefrequencyA.csv and allelfrequencyB.csv – example format for allele frequency distribution files for 10 loci and 5 alleles.
    - classvars/
        - ClassVarsA.csv and ClassVarsB.csv – example demographic files with parameters/variables for age/size classes
    - PopVars.csv – run and population parameters corresponding to the example files
    - PatchVars.csv – example file with parameters/variables for each patch
    - PatchVars_climate.csv – example file for how to implement CDClimate module at the patch level.
    - Patch_r.csv – example correlation matrix for 8 parameters.

2.3 Example run

2.3.1 Command line run

The example run is for 7 patches representing individuals with an effective distance matrix calculated using a least-cost path algorithm through riverine distance (Fig. 6).

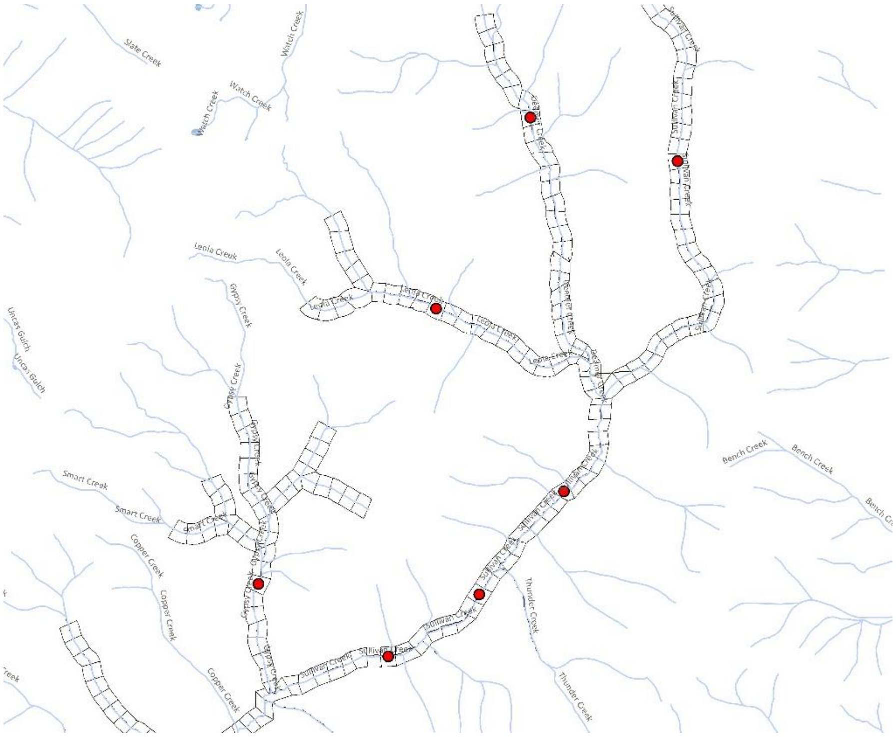


Figure 6: Example riverine system with 7 patches centroids as red. Variables and parameters correspond to each patch location (PatchVars.csv).

To run the following example, follow these steps:

1. Double check that the 3 directories provided in the archive are in the same directory.

2. The included file 'PopVars.csv' in the data directory specifies the parameters that can be changed and used in a sample CDMetaPOP run. Open 'PopVars.csv' in your editor of choice. A spreadsheet program like Microsoft Excel, allows for easy editing of the tabular values.

3. The file 'PopVars.csv' defines the patch files in the first column. The included files 'PatchVars.csv' will also be in the same folder (../data). 'ClassVars.csv' files are in turn specified in the 'PatchVars.csv' file and example 'ClassVars.csv' will be in the ../data/classvars/ folder.

4. There will be 3 lines of information in 'PopVars.csv': a header line and 2 lines of information corresponding to 2 separate CDMetaPOP runs (batch process). See Table 1 which contains a breakdown for each column header and the parameters that can be changed. The 'Input' in the table listed is for the first row in the file. Make sure you save this file in the same format – a comma delimited file – when you make changes to the parameters. Do not change the 'Input' (first row) labeling. Select 'Yes' or 'OK' for any Excel questions about saving in this format.

5. Start the program: For example, if you use python from the command line, then open a terminal window and change your shell directory to the CDMetaPOP src home directory (i.e., > cd C:\”homedirectorylocation”\src).

6. Run the program: There are a number of ways to run this program. If you are using a command shell you can run the program by typing “python CDMetaPOP.py C:/”homedirectorylocation”/data PopVars.csv output_test”. Or a short-cut if your data is located at the same folder level as the src folder: “python CDMetaPOP.py ../data/ PopVars.csv output_test”. Note that there are 5 arguments here that must be included with spaces in between:

   - “python” starts python, for example from the command line. Note that other python environments may have different calls here. In PyLab (the IDE distributed with EPD), the call is “run”.
   - “CDMetaPOP.py” runs CDMetaPOP program.
   - “C:/”homedirectorylocation”/data” is the directory location of the input test files. You can point this directory to other project files, for example. We suggest not having any spaces in your directory names. So as projects accumulate you can rename input folders that contain the project specific files (e.g., dataWestslope or dataBullTrout).
   - “PopVars.csv” is the parameter file (comma delimited) which can be renamed (e.g., “PopVars_WCT.csv”). Caution should be taken when going between operating systems and saving this file as a .csv.
   - “output_test” is the name of the directory that will be created with CDMetaPOP output in the directory specified by the third argument above.

Check for successful model run completion: The program will provide step-by-step output in the Shell window. Once completed, a simulation time will be printed out and folders batchrun0mcrun0, batchrun0mcrun1, batchrun0mcrun2, batchrun1mcrun0, and batchrun2mcrun0 will be created in your CDMetaPOP home directory to store output from the separate batch and/or Monte-Carlo runs (each line in the PopVars file corresponds to a separate batchrun and the specified ‘mcruns’ for each batch). These folders are located in the data folder specified in above step. The output folder will have a unique date/time stamp after the name of the output folder in case you want to run multiple CDMetaPOP runs in this same directory. The program will also provide a log file with program steps in your specified output directory. If parameters are such that population becomes extinct before specified generation time, then program will end. The program will provide error and feedback for parameters that are outside of ranges or incorrectly entered.

3 Input

This section lists the specific information for each input file and corresponding parameters used in CDMetaPOP. See examples provided for formatting (PopVars.csv, PatchVars.csv, and ClassVars.csv). The file headers listed are for the first row in each input file describing each file or parameter (do not modify this line in the input files).

3.1 Patch level controls – PatchVars.csv file

See PatchVars.csv as the example input file for patch level controls. An easy way to create this file is to establish first X,Y locations for each patch and develop raster surfaces for each variable (temperature, grow days, fitness values, and mortality values linked to climate or harvesting spatially). Then use a software program ('raster' in R or ArcGIS) to extract variables under each X,Y patch location, such as temperature, grow days, and fitness values. Through these software programs, additional fields can be added to help populate K, N0, or Gene Initialization.

File Header in .csv	Example	Description
xyfilename	'PatchVars.csv' – see example supplied for 7 patches.	The *n*-(*x*,*y*) patch locations with 28 columns/fields of information for each patch. This is a comma delimited file. Some fields are optional and will not be considered – leave default values. * parameters can vary through time either systematically or through temporal variation. Parameters with '_Std' can be used for temporal variation and to create a covariance matrix with selected parameters (e.g., correlation_matrix: Patch_r.csv). See CDClimate section for details on how to implement systematic temporal variation. (*Patch*)- a unique identifier for each patch. Begin label 1 through *n* in consecutive order.

		(*X*)-x-coordinate location of patch. (*Y*)-y-coordinate location of patch. *(*K*)-the carrying capacity for each patch. A value of 0 can be included here and individuals will not disperse (migration out or stray) to these patches. *(*K_Std*)-the standard deviation on K for each patch which will vary patch K values through time. If a constant K per patch is wished, then enter 0 here. (*N0*)-corresponding patches will be initialized with this number of individuals. (*Natal Grounds*)-designates natal ground locations. '1' – individuals can occupy this patch for e.g., spawning. '0' – individuals cannot occupy this patch when they are 'back'. N0 cannot be > 0 (initialization value starting individuals at natal grounds), if natal ground patch is 0. (*Migration Grounds*)-designated migration ground locations. '1' – individuals can occupy this patch during DoEmigration() process (e.g., overwintering). '0' individuals cannot occupy this patch when they are 'out'. (*Genes Initialize*)- The choice for how to initialize the genotype for each

		individual. Different initialization can be used for each patch. • Use 'random' to get a random assignment of alleles (number of alleles and loci specified in PopVars.csv). Patch will be at maximum genetic diversity. • Enter an allele frequency file name: Then the alleles are drawn from the allele frequency distribution in file. Examples of format in comma delimited form are given in the data/genes/ folder location. This file is a column of allele frequencies and make sure the length of the column matches starting loci * starting alleles given in the PopVars.csv file. Different allele frequency files can be given for different patches, making sure they are the same size (loci * alleles). (*Class Vars*)-the ClassVars input file that governs this patch location. See more on ClassVars specific parameters in next section 3.2. *(*Mortality Out %*) – This is the constant mortality percentage [0-100] applied to each patch in the rearing/overwintering/foraging stage. • Enter 'N' here and this field is not considered. Note this is compounded with age and size level mortalities with given constMortans. • In the special case in which e.g., eradication as well as suppression operating at the class level is desired, then enter 'E'. This will override any class level mortality values (described below).

		*(*Mortality Out StDev*) – This is the standard deviation around the ‘Mortality Out %’ [0-100] applied to each patch. Enter 0 here and this field will be deterministic, using the ‘Mortality Out %’ value at each time step. *(*Mortality Back %*) – This is the constant mortality percentage [0-100] applied to each patch after immigration back to original/natal population. - Enter ‘N’ here and this field is not considered. Note this is compounded with age and size level mortalities with given constMortans. - In the special case in which e.g., eradication as well as suppression operating at the class level is desired, then enter ‘E’. This will override any class level mortality values (described below). *(*Mortality Back StDev*) – This is the standard deviation around the ‘Mortality Back %’ [0-100] applied to each patch. Enter 0 here and this field will be deterministic, using the ‘Mortality Back %’ value at each time step. *(*Mortality Eggs %*) – This is the egg or Age 0 mortality percentage [0-100]. Note that there is also a population wide level consideration on egg mortality (see below) in PopVars file. If used together, events will be considered mutually exclusive (additive). *(*Mortality Eggs StDev*) – This is the standard deviation around the ‘Mortality Eggs %’ [0-100] applied to each patch. Enter 0 here and this field will be deterministic, using the ‘Mortality Eggs %’ value at each time step.

		*(*Migration*) – The emigration probability [0-1] applied before moving to rearing/overwinter grounds. If just age/size class migration is wanted, then set these values to 1. (*Set Migration*) – If an individual becomes a migrant, it can stay a migrant with probability 1.0. • To turn this on, specify 'Y'. • To turn this off, specify 'N'. *(*Straying*) – The local dispersal/straying probability [0-1] applied before moving back to original/natal grounds. Set these values to 1 and age class can govern through Classvars. *(*Growth Temperature Out*) – Temperature values that influence body size growth of individuals at patch location after DoEmigration() or migration out of natal grounds (e.g., during fall/winter for a spring breeder). • Extract temperature values spatially to patch locations, or • Enter 'N' for individual patches or all, which can turn off temperature dependent patch specific growths at this time of year. *(*Growth Temperature Out StDev*) – This is the standard deviation around each temperature value 'out' and applied to each patch. Enter 0 here and this field will be deterministic, using the 'Growth Temperature Out' value at each time step.

		*(*Growing Days Out*) – Patch site growing days entered here corresponding to the growth time 'out' of natal grounds in foraging patches for growth option 'temperature'. If 'N' is used in the 'Growth Temperature Out', then this value is ignored. *(*Growth Days Out StDev*) – This is the standard deviation around each grow days 'out' and applied to each patch. Enter 0 here and this field will be deterministic, using the 'Growth Days Out' value at each time step. *(*Temperature Back Growth*) – Temperature values that influence body size growth of individuals at patch location after DoImmmigration() or migration back to natal grounds (e.g., during summer for a spring breeder). • Extract temperature values spatially to patch locations, or • Enter 'N' for individual patches or all, which can turn off temperature dependent patch specific growths at this time of year. *(*Growth Temperature Back StDev*) – This is the standard deviation around each temperature value 'back' and applied to each patch. Enter 0 here and this field will be deterministic, using the 'Growth Temperature Back' value at each time step. *(*Growing Days Back*) – Patch site growing days entered here corresponding to the growth time 'back' at natal grounds. If 'N' is used in the 'Growth Temperature Back', then this value is ignored. *(*Grow Days Back StDev*) – This is the standard deviation around each grow day value 'back' and applied to each patch.

		Enter 0 here and this field will be deterministic, using the 'Grow Days Back' value at each time step. *(*Capture Probability Back*) – the probability of detection, capture, or sampling for each patch when individuals are back at natal grounds (e.g., for spring spawning fish during the summer). This occurs right before the individuals are preparing to migrate out of their natal grounds. • Enter a probability [0-1] or • Enter 'N' to not consider this option. • Enter 1 and class level capture probability will operate (ClassVars). *(*Capture Probability Out*) – the probability of detection, capture, or sampling for each patch when individuals are away from their natal grounds (e.g., for spring spawning fish during the winter). This occurs right before the individuals are preparing to migrate back to their natal grounds. • Enter a probability [0-1] or • Enter 'N' to not consider this option. • Enter 1 and class level capture probability will operate (ClassVars). *(*Fitness_AA, Fitness Aa, Fitness_aa*) – When CDEVOLVE answer is 1 (in PopVars.csv file), then this is the offspring viability selection values for AA, Aa, and aa, respectively (i.e., one-locus selection model). These are differential mortality values tied to each genotype. You can link the selection via mortality

		to spatial environmental-genotype processes. When CDEVOLVE answer is 'M' or 'MG' (in PopVars.csv file), then these 3 genotypes are used to enter 4 maturation response parameters in the order male_slope;male_intercept;female_slope;female_intercept. The 4 values must be entered and separated by a ';'. CDCLIMATE can still be considered, e.g., 0.06;23;0.08;21|0.08;24;0.10;22 using the '|' to separate each parameter group to be considered. *(*Fitness_BB, Fitness Bb, Fitness_bb*) – When CDEVOLVE answer is 'G' or 'MG' (in PopVars.csv file), then these 3 genotypes are used to enter 5 growth rate response parameters in the order temperature;353;0.57;13;0.33;-0.196. The 4 values must be entered and separated by a ';'. CDCLIMATE can still be considered, e.g., temperature;353;0.57;13;0.33;-0.196| temperature;353;0.57;13;0.33;-0.196 using the '|' to separate each parameter group to be considered. *(*Fitness_AABB, Fitness AaBB, Fitness_aaBB, Fitness AABb, Fitness AaBb, Fitness aaBb, Fitness Aabb, Fitness Aabb, Fitness aabb*) – When CDEVOLVE answer is 2, then this is the offspring viability selection values for each of the 9 genotypes (i.e., two-locus selection model). These are differential mortality values tied to each genotype. You can link the selection via mortality to spatial environmental-genotype processes. For example, to consider one genotype under directional selection, e.g., AABB, enter spatial mortality values here [0-100] and all other genotypes having 0% mortality.

3.2 Class level controls – ClassVars.csv file		
See ../classvars/ClassVarsA.csv as an example input file for the age/size parameters and formatting (e.g., comma delimited with the below headings). These files determine age/size class information for the program. Supply a file in the /data/ folder. However, input file locations are specified in the PatchVars.csv input file (see above). Hence, a separate ClassVars input file can be specified for each patch if wished.		
Agefilename	‘ClassVarsA.csv’ – an example age and size class file with 6 classes.	Age/size class information for the program with the following headings. (*Age class*) – Number of age classes by sequential class numbers – 0, 1, 2, … (*Body Size Mean*) – the mean body size for this age class, e.g., length in mm. If size control is specified, then individuals are grouped around these size classes specified here and associated parameters in the rest of this file used. In addition, if ‘known’ growth option is specified, then these values will be used as the size at given age. (*Body Size Std*) – the standard deviation in body sizes for this age class. Together with ‘Body Size Mean’, these numbers initialize the body size for the starting individuals from a random draw from a uniform distribution with mean and standard deviation. (*Distribution*) – numbers here specify the distribution within each class. (*Percent Female*) – this is for initializing the sex of the population which can give a ratio of females to males per age class.

		• A special case for Wright Fisher assumption can be specified here by entering 'WrightFisher' and only should be used when considering a panmictic population (see section on *Special Cases* for more details here). (*Age Mortality Out* %) – the age specific mortality percentages [0-100] applied in overwintering/rearing/foraging patches – occurs after DoEmigration() module. Enter 'N' here and this will be ignored. Note this is compounded with patch and size level mortalities with given constMortans. (*Age Mortality Out StDev*) – the standard deviation for the age specific mortality percentages [0-100] applied in 'out' phase. Enter 0 here and this parameter, 'Age Mortality Out', will be deterministic with the value entered in 'Age Mortality Out' field used at every time step. (*Age Mortality Back* %) – the age specific mortality percentages [0-100] applied in original/natal population – occurs after DoImmigration() module. Enter 'N' here and this will be ignored. Note this is compounded with patch and size level mortalities with given constMortans. (*Age Mortality Back StDev*) – the standard deviation for the age specific mortality percentages [0-100] applied in 'back' phase. Enter 0 here and this parameter, 'Age Mortality Back', will be deterministic with the value entered in 'Age Mortality Back' field used at every time step. (*Size Mortality Out* %) – the size specific mortality percentages [0-100] applied in overwintering/rearing/foraging patches –

		occurs after DoEmigration() module. Enter 'N' here and this will be ignored. Note this is compounded with patch and age level mortalities with given constMortans. (*Size Mortality Out StDev*) – the standard deviation for the size specific mortality percentages [0-100] applied in 'out' phase. Enter 0 here and this parameter, 'Age Mortality Out', will be deterministic with the value entered in 'Age Mortality Out' field used at every time step. (*Size Mortality Back %*) – the size specific mortality percentages [0-100] applied in original/natal population – occurs after DoImmigration() module. Enter 'N' here and this will be ignored. Note this is compounded with patch and age level mortalities with given constMortans. (*Size Mortality Back StDev*) – the standard deviation for the size specific mortality percentages [0-100] applied in 'back' phase. Enter 0 here and this parameter, 'Age Mortality Back', will be deterministic with the value entered in 'Age Mortality Back' field used at every time step. (*Migration*) – the class probability [0-1] of migrating out of original/natal population during the DoEmigration() module. Set these values to 1 for patch level control on migration (see PatchVars.csv). (*Straying*) – the class probability [0-1] of locally dispersing to all patches during the DoImmigration() module. Set these values to 1 for patch level control on straying (see PatchVars.csv).

		(*Male Maturation*) – the probability of being a reproducing individual and then stay a reproducing individual. If size option is specified, then these values are not used and population fit parameters based on size/length relationships are used (PopVars.csv). (*Female Maturation*) – the probability of being a reproducing individual, and then stay a reproducing individual. If size option is specified, then these values are not used and population fit parameters based on size/length relationships are used (PopVars.csv). (*Fecundity (mean birth number and standard deviation)*) – the litter size or egg number with standard deviation option. Poisson, random, constant, or normal draw options available (in PopVars.csv file). If size option is specified, then these values are not used and population fit parameters based on size are used (PopVars.csv). (*Capture Probability Back*) – the probability of detection, capture, or sampling for each class when individuals are back at natal grounds (e.g., for spring spawning fish during the summer). This occurs right before the individuals are preparing to migrate out of their natal grounds. • Enter a probability [0-1] or • Enter ‘N’ to not consider this option.

		• Enter 1 and patch level capture probability will operate (PatchVars.csv). (*Capture Probability Out*) – the probability of detection, capture, or sampling for each class when individuals are away from their natal grounds (e.g., for spring spawning fish during the winter). This occurs right before the individuals are preparing to migrate back to their natal grounds. • Enter a probability [0-1] or • Enter ‘N’ to not consider this option. • Enter 1 and patch level capture probability will operate (PatchVars.csv).
Sizecontrol	‘Y’ or ‘N’	If ‘Y’ is specified, then body size will drive processes that are linked to that size class, e.g., in ClassVars.csv file – migration, straying, mortality – in PopVars.csv file – fecundity, maturation and growth parameters. E.g., for size 83 mm, all individuals will be sorted and those closest in size to 83 mm will be binned together. Then the mortality, migration, and straying values for row in input file with size 83 mm will be used for those individuals. Tracking numbers (see Section 4) are then summarized around these size class bins. If ‘N’ is specified, the age will drive processes linked in ClassVars.csv file. Tracking numbers (see Section 4) are then summarized around the age classes.

3.3 Run parameters and output – PopVars.csv file

See PopVars.csv for an example input file. This file 'runs' the program, specifying program controls as well as population level parameters.

constMortans	'1' or '2'	This option applies how the patch, size and/or age constant mortality value events are considered and compounded. • '1' – refers to mutually exclusive events or mortality percentages are added (i.e., P(A) + P(B)). • '2' – refers to independent events or mortality percentages are multiplied.
mcruns	'5' – 5 replicate runs denoted in folders labeled with 'mcrun0', 'mcrun1',…	The repeated number of simulations to be conducted for the Monte Carlo method (i.e., the number of replicates for 1 batch of parameters). Rows in the 'PopVars.csv' file indicate a new simulation run or batch run. Labeling begins counting with 0. In the output folder, and example with mcruns = 2 with 3 rows of parameters would give the following folder output: batchrun0mcrun0, batchrun0mcrun1, batchrun1mcrun0, batchrun1mcrun1, batchrun2mcrun0, batchrun2mcrun1
runtime	'10' – 10 time steps	Simulation run time [generation or year]. File output indexed from 0 – (runtime-1). For example ind0.csv, ind1.csv, ind2.csv, ind3.csv, and ind4.csv would be outputted for a runtime of 5.
output_years	'0', '1', '0|3|4' – the sequence value or list of generations to save.	The specified simulation time steps [year/generation] to write to file, summarize year metrics, or to calculate genetic distance matrices. • Enter a single number to produce a sequence of values. For example '1' with runtime = 10 would produce

		output for years 0, 1, …, 9. A value of '2' with runtime = 10 would produce output for years 0, 2, 4, 6, 8. • Enter exact years by using a '|'. For example '0|3|4' would produce output for years 0, 3, and 4. Note that years begin counting at 0, so the last value must be one less than the runtime (e.g., runtime = 10, then 0|5|9 with 9 being the maximum value for the last year).
Gridformat	'genepop', 'genalex', 'structure', 'genepop' or 'cdpop'	This is the genes output format option: • 'cdpop' – This format is the default. The cdpop format lists the genotypes with values for each allele; either 0, 1, or 2. Output will be labeled ind{year}.csv. Note that this format can now be read into many population genetics programs (e.g., PopGraph in R (Dyer 2005)). • 'general' – for a general genotype output. The general format will follow Locus1a, Locus1b, Locus2a, Locus2b, …, LocusNa, LocusNb. Output will be labeled general_ind{year}.csv • 'genalex' – for the program GENALEX (Peakall and Smouse 2006, 2012). Output will be labeled genalex_ind{year}.gen • 'structure' – for the program STRUCTURE (Pritchard et al. 2000). Output will be labeled structure_ind{year}.stru

		- ‘genepop’ – for the program GENEPOP (REF) or related programs. Output will be labeled genepop_ind{year}.gen
gridsampling	‘N’ or ‘Sample’	An option to output genotypes before the Immigration step or when they are away from their natal grounds. - Enter ‘N’ for output when individuals are at their natal grounds. E.g., ind0.csv, ind1.csv, … - Enter ‘Sample’ for output when individuals are away from their natal grounds. Two files for each year specified will be seen in the output file: e.g., ind0.csv, indSample0.csv, ind1.csv, indSample1.csv, …
gendmatans	‘N’ or ‘Dps’	Option to calculate a genetic distance matrix for each individual at the specified output year. Note, that there are a number of metrics and programs to calculate genetic distances and at this time we only provide one option. We do not advise using this option when over 1000 individuals due to computational time. - Enter ‘N’ to not calculate a matrix. - Enter ‘Dps’ to calculate a matrix using proportion of shared alleles (Bowcock et al. 1994).
summaryOutput	‘Y’ or ‘N’	Option to output summary metrics for each patch at the given time intervals. See Section 4 on output metrics. Depending on the size of the system and simulation time, these summary output files can be very big, so consider time and space

		limitations on your local machines when selecting this answer. - Enter 'Y' to produce output summary metrics for each time interval and patch. - Enter 'N' to not produce output metrics.
Popmodel	'N', 'logistic', 'packing'	The choice of population growth models. - Enter 'N' – this population model is essentially exponential growth where $N_j(t+1)$ = birth-rate * $N_j(t)$ – death-rate * $N_j(t)$. Population numbers can reach the set carrying capacity for each patch, but not exceed this number. This model is considered for all patches after migration out and after migration back. If $N_j > K_j$, then a random sample from the patch is removed. - Enter 'packing' to consider the density-dependent class specific population model. - Enter 'logistic' to consider density dependent age/stage structured model following the logistic equation.
Popmodel_par1	-0.6821	The parameter used for the packing algorithm.
correlation_matrix	Filename or 'N'	This is the file name for the correlation matrix used to generate correlated variables at the patch level. Currently, there are 8 parameters in which the user can specify correlation structure: K, popmort_out, temperature_out, growdays_out, popmort_back, temperature_back, growdays_back, eggmort. See 'Patch_r.csv' for the order and example of each potential correlated

		variable. Enter 'N' and the covariance matrix will not be used.

3.4 CDClimate and movement surfaces with functions

See PopVars.csv first batch row and PatchVars_climate.csv as an example for how to implement systematic changing variables and surfaces. Currently the variables that can be changed include

- *Movement surfaces, functions, and thresholds (in PopVars.csv input and described below)*
- *K (in PatchVars.csv input)*
- *Mortality at the patch control (in PatchVars.csv input)*
- *Migration and straying probabilities at the patch control (in PatchVars.csv input)*
- *Patch specific temperature and grow days for two times of the year (e.g., summer and winter) (in PatchVars.csv).*
- *Spatial selection surfaces for genotype-environment, -maturation, and -growth processes (in PatchVars.csv and described in Section 3.8)*

cdclimgentime	'0|5|10' or '0'	To initiate the CDClimate module, this is the generation or year that the next variable or effective distance matrix will be switched. • Place only a '0' here to start simulations with one surface (and initial parameters) and continue to use throughout runtime. This is the default value. • You can specify multiple years by separating each generation by '|'. Then in the following surface columns (described immediately below), a separate file can be given for each generation.
Mate_cdmat	'Patch7_Cdmatrix.csv' – an example effective distance matrix used for the	A *[n x n]* effective distance matrix for mating movement, where *n* is the number of patches on the landscape. This is a comma delimited file. See the example given for formatting this file in the data/cdmat/ folder. This file can be calculated from any program you choose (e.g., PATHMATRIX,

	mating movement.	CIRCUITSCAPE, UNICOR, COSTDISTANCE, and with R gdistance functions). This file can be asymmetrical (see gdistance package for calculating asymmetrical cost) and the order is assumed to read columns, e.g., costVal[1,2] > costVal[2,1] if for example, patch 1 is higher in elevation than patch 2. When using asymmetrical cost, then double check that this is in the correct column order. If the CDClimate module was initiated with multiple years in the field 'cdclimgentime', then the same number of surfaces must be given here separated by a '|', e.g., 'Patch7_Cdmatrix.csv|Patch7_Cdmatrix.csv|Patch7_Cdmatrix.csv' could correspond to the example given for reading in a new surfaces at generations '0|5|10'. Note that you can place these files in a separate folder, just make sure that the field still points to the correct files.
Dispout_cdmat	'Patch7_Cdmatrix.csv'	Used for the movement out of each subpopulation. Same description as 'mate_cdmat'. It also can be the same file as the mate, dispersal back or straying effective distance matrix.
Dispback_cdmat	'Patch7_Cdmatrix.csv'	Used for the movement back to natal subpopulation. Same description as 'mate_cdmat'. It also can be the same file as the mate, dispersal out or straying effective distance matrix.
Stray_cdmat	'Patch7_Cdmatrix.csv'	Used for the movement of local dispersal or straying to other patches. Same description as 'mate_cdmat'. It also can be the same file as the mate, dispersal out or dispersal back effective distance matrix.

		Here, in order to stray individuals randomly, use the movement function answer ‘4’ below. You can use a matrix of 0s and 1s to control for random dispersal while neglecting some patches.
Matemoveno, FdispmoveOutno, MdispmoveOutno, FdispmoveBackno, MdispmoveBackno, StrayBaackno	‘6’ – Movement function answer for probability.	You can additionally control the probability of movement and effective distance distribution through functions. All probabilities are scaled between 0 and 1. Some functions below are naturally between 0-1, while others use the minimum, maximum, and threshold values of the effective distance matrix to rescale. • ‘1’ = Linear: probability = (1 – (1/Threshold) * Effective Distance) • ‘2’ = Inverse Square: probability = (1 / (Effective Distance^2)). This function gets rescaled between 0-1 using the min and threshold of the inverse square effective distance. • ‘3’ = Nearest Neighbor: Moore neighborhood random neighbor chosen. Often the case however, is if patches are not on a uniform grid, then the closest neighbor will be chosen. • ‘4’ = Random Mixing: Will consider the cost distance threshold. Use the maximum effective distance in the threshold field if you want to consider the entire population as random movement. • ‘5’ = Negative Exponential: probability = (A * 10^(-B * Effective Distance)). This function

		gets rescaled between 0-1 using the min and threshold of the negative exponential effective distance. • ‘6’ = Random Patch: Given the patch *j*, then movement will occur only within *j*. The function could be used to test defined patches with no migration between. • ‘7’ = Gaussian function: probability = A * exp (- (Cost Distance – B)^2 / (2 * C^2)). This function gets rescaled between 0-1 using the min and threshold of the Gaussian effective distance. • ‘8’ = Use the cost distance matrix: probability = 1 – the cost distance matrix rescaled to the min and threshold. • ‘9’ = Use a probability matrix: no function is applied to values and the straight probability matrix is used. Values must be from 0 – 1. If ‘F’ or ‘M’ is before this field name, then separate options can be entered for female or male movement, controlling for biased dispersal.
matemoveparA, FdispmoveOutparA, MdispmoveOutparA, FdispmoveBackparA, MdispmoveBackparA,StrayBackparA	‘0.0005’	This is the A parameter used for the function answer ‘5’, ‘7’. Specify same number of values here if CDClimate is initiated, e.g., 0.05|0|0.
matemoveparB, FdispmoveOutparB, MdispmoveOutparB,	‘0.01’	This is the B parameter used for the function answer ‘5’, ‘7’. Specify same

FdispmoveBackparB, MdispmoveBackparB,StrayBackparB		number of values here if CDClimate is initiated, e.g., 0.05|0|0.
matemoveparC, FdispmoveOutparC, MdispmoveOutparC, FdispmoveBackparC, MdispmoveBackparC, StrayBackparC	‘0.01’	This is the C parameter used for the function answer ‘7’. Specify same number of values here if CDClimate is initiated, e.g., 0.05|0|0.
Matemovethresh, FdispmoveOutthresh, MdispmoveOutthresh, FdispmoveBackthresh, MdispmoveBackthresh,StrayBackthresh	‘max’, ‘%max’, or cost units	A threshold option (in effective distance units) for how far an individual can search for a mate (i.e., mate selection), migrate, or disperse to a new patch location, equivalent to the effective distance kernel. • Enter ‘max’ to consider all patches for movement or all individuals for mate selection. • Enter ‘%max’ to consider a percent cost distance movement for mating. The % is an integer value between 1 and 100. For example ‘10max’ would consider all mating individuals that are within 10 percent of the maximum cost distance on the surface. Caution using this option when comparing across landscape surfaces. • Enter a value in cost distance units to consider movement within a set threshold.
HomeAttempt	‘mortality’, ‘stray_emiPop’, or ‘stray_natalPop’	There is a possibility that a migrant that did not become a strayer attempts to immigrate back to its original natal patch but cannot. This can occur either because K = 0 at that original natal patch location, e.g., a catastrophic event, or the cost to move back to the original natal patch exceeds the specified movement threshold (e.g., an individual emigrated too far perhaps down over a barrier). The

		following options are available for modeling these special case individuals: • ‘mortality’ – the individual is assumed to die. • ‘stray_emiPop’ – the individual will attempt to stray to accessible patch locations using the stray matrix movement rules and from it’s emigrated patch. Note that there still is a possibility that the stray matrix rules will produce inaccessible locations in which case the individual will assume to die. • ‘stray_natalPop’ - the individual will attempt to stray to accessible patch locations using the stray matrix movement rules and from it’s natal patch. The difference here and rational is that the individual wants to stray to a patch closest to it’s natal location. A recursive operation is used to check to make sure there are indeed patch locations in which probability is greater than 1 to ensure that an individual does not travel up over a barrier. Note that there still is a possibility that the stray matrix rules will produce inaccessible locations in which case the individual will assume to die.
3.5 Reproduction options		
sexans	‘Y’ or ‘N’	This option determines heterosexual or asexual reproduction • Enter ‘Y’ for sexual reproduction. In sexual reproduction, mated pairs consider male and females with or without replacement. • Enter ‘N’ for asexual reproduction. With asexual reproduction, all mature individuals are considered for mating with mates selected according to the

		movement function choice and without regard to any gender or mating type. It is important to note that this “asexual” reproduction is functionally the sexual paring of hermaphroditic individuals; thus it is asexual in the sense that there are no distinct sexes, but is sexual in the sense that individuals mate with other individuals, exchanging genes in Mendelian reproduction.
selfans	‘Y’ or ‘N’	Consider selfing, i.e., individuals exchange genes with self. - Enter ‘Y’ to consider selfing - Enter ‘N’ to turn off selfing.
Freplace, Mreplace	‘Y’ or ‘N’	These options determine the mating structure, i.e., monogamy, polygamy, or polyandry. - Enter ‘Y’ to consider females or males to mate with replacement. - Enter ‘N’ to consider females or males to mate without replacement.
mature_length	Size, age, or ‘N’	Use a set length (for size control) or set age (for age control) to specify when a female and male becomes mature. This will be a deterministic process. Enter ‘N’ here to use a length-probability stochastic process with parameters specified below. * Note that there is a check on maturity, i.e., if an individual reaches age 6, the program assumes that it will be mature if not already.

mature_slope, mature_intercept		If the size control is specified, then these are the parameters that fit the probability of maturing as a function of size. These values can be modeled separately for males and females. The function used is from Downs et al. 1997: P(mature) = exp(mature_intercept + mature_slope * Length) / (1 + exp(mature_intercept + mature_slope * Length)) If age control is operating, then these values are ignored and the probability values entered in the ‘ClassVars.csv’ file are used. Once an individual becomes mature, then it stays mature.
3.6 Offspring/litter/egg options		
offno	‘1’, ‘2’, ‘3’, or ‘4’	This is the draw for number of offspring (e.g., eggs or litter or clutch size) that each mate pair will have. If age control is specified, then use the fecundity mean and standard deviation values in the ‘ClassVars.csv’ file for the corresponding draw choice below. If size control is specified, then the parameter options in the PopVars.csv file will determine the mean fecundity value used (described in more detail next). • ‘1’ – for a random draw between 0 and mean fecundity value. • ‘2’ - for Poisson draw of mean fecundity value. • ‘3’ - for a constant number of offspring of mean fecundity value.

		• ‘4’ - for a normal draw with mean fecundity and standard deviation.
equalClutchSize	‘Y’ or ‘N’	This option determines equal clutch size per mate. • Enter ‘Y’ for an equal clutch size for each female of mean fecundity value resulting from ‘offno’ choice above. • Enter ‘N’ for an unequal clutch size for each female. For example, if a female mates with 4 males and ‘offno’ = 3 with fecundity value of 2, then she will have 8 offspring. For example, if a female mates with 4 males and ‘offno’ = 3 with fecundity value of 2, then with an equal clutch size (‘Y’) she will have 2 offspring total and the fathers and resulting genes are randomly selected from the 4 mating events. She will have 8 offspring (2 from each mate event) with ‘N’ entered.
eggFrequency	0 - 1	This is the frequency (count per year) in which the female will lay eggs. This value ranges between 0 – 1. E.g., 0.5 would mean that a mature female will lay eggs every other year.
Egg_Mean_ans	‘exp’, ‘linear’, or ‘pow’	If size control is specified, then this function choice is used to produce the number of offspring a female will have. • ‘exp’ – E_mu = Egg_Mean_par1 * exp (Egg_Mean_par2 * Size) • ‘linear’ - E_mu = Egg_Mean_par1 + Egg_Mean_par2 * Size

		• *‘pow’ - E_mu = Egg_Mean_par1 * Size ^ Egg_Mean_par2*
Egg_Mean_par1, Egg_Mean_par2,	-790, 6.2	If size control is specified, then these are the parameters values used to fit size based fecundity function above.
Egg_Mortality, Egg_Mortality_StDev	62, 18	Mortality of eggs will occur around value with given standard deviation. If a nonzero value is placed in ‘Egg_Mortality_StDev’, then stochastic variability will occur every time step (a new value will be drawn around ‘Egg_Mortality’ with given standard deviation. Note this can be compounded with the use of the patch level egg mortality and events will be considered mutually exclusive.
Egg_FemalePercent	50	This option can be used to consider unequal sex ratio at birth (for eggs) and is the percent females born in litter (100 – ‘Egg_FemalePercent’ gives the percent males born in the litter). • A special case for Wright Fisher assumption can be specified here by entering ‘WrightFisher’ and only should be used when considering a panmictic population (see section on *Special Cases* for more details).

3.7 Genetic options

startGenes	0 - runtime	The year/generation at which genetic exchange will begin. For example, use a later year to begin swapping genes while population dynamics is still stabilizing.
Loci	2 – max number dependent on	The number of loci (microsatellites). Recommended maximum number of loci

	computer resources.	dependent on computer resources and recommend test runs for tradeoffs in performance. If a file is specified in the ClassVars.csv input file, then the number of loci entered here must match this file.
Alleles	2 – max number dependent on computer resources.	The number of starting alleles per locus. If a file is specified in the ClassVars.csv input file, then the number of alleles entered here must match this file. Polymorphism or varying number of alleles can be used by specifying the maximum number of alleles here and filling in allele frequency values of 0 for ‘filler’ alleles in other loci locations.
Muterate	0 - 1	The allele mutation rate.
Mutationtype	‘random’, ‘forward’, ‘backward’, ‘forwardbackward’	The type of mutation model: • ‘random’ – This is the kth-allele mutation model. • ‘forward’ – This is a step-wise mutation in which an allele can mutate forwards only (i.e., to the right). • ‘backward’ – This is a step-wise mutation in which an allele can mutate backwards only (i.e., to the left). • ‘forwardbackward’ – This is a step-wise mutation in which an allele can mutate forward or backwards only (i.e., to the left or right with equal probability).

		- ‘forwardAbackwardBrandomN’ – This is a special case for the 2-loci selection model. The first locus under selection can only go forward (A -> a) and the second locus under selection can only go backward (b -> B). The rest of the neutral loci are random mutations.
mtdna	‘Y’ or ‘N’	An option for tracking maternal genes: - Enter ‘Y’ and the last locus becomes the maternal marker (mtDNA) and every offspring inherits this locus from its mother only. - Enter ‘N’ and regular Mendel inheritance occurs for this last locus.

3.8 Spatial selection options

CDevolveans	‘N’, ‘1’, ‘2’, ‘M’, ‘G’, ‘MG_ind’, ‘MG_link’, ‘1_mat’, ‘2_mat’	This is the answer for how many loci are under selection. Currently as implemented, alleles must be 2 for selection models. - Enter ‘N’ to turn off CDEVOLVE. - Enter ‘1’ for selection with 1 locus A. - Enter ‘2’ for selection with 2 loci (A and B). - Enter ‘M’ for Locus A (first locus) tied to maturation probability curves. - Enter ‘G’ for Locus B (second locus) tied to growth curves.

		• Enter ‘MG_ind’ for both maturation and growth independently linked to Locus A and Locus B, respectively. • Enter ‘MG_link’ for both maturation and growth linked to Locus A. E.g., If AA genotype, then the respective maturation parameter values are used for this genotype and linked to growth values listed for BB genotype. Aa is linked to Bb, and aa is linked to bb. • Enter ‘1_mat’ to consider selection with 1 locus (A) with only mature individuals. • Enter ‘2_mat’ to consider selection with 2 loci (A and B) with only mature individuals. Spatial selection values (differential mortality for options ‘1’ and ‘2’ and parameters values for options ‘M’ and ‘G’) for each selection model and corresponding genotype are then entered for each patch (see Patch control and PatchVars.csv file).
startSelection	0 – runtime	This is the time unit that the selection surface will begin operating on loci under selection, specified in previous field (cdevolveans). If ‘N’ is specified for ‘cdevolveans’, then this field is ignored. Note that ‘startSelection’ must be >= ‘startGenes’.
implementSelection	‘Out’, ‘Back’, ‘Both’	The option to apply selection surfaces:

		- ‘Out’ – when individuals are migrated out and away from natal grounds. - ‘Back’ – when individuals are back at natal grounds. - ‘Both’ – apply spatial selection surface at both timing events.
3.9 Infection options		
Cdinfect	‘Y’ or ‘N’	This option tracks vertical transmission in the population. A column in ind{}.csv denotes the infection status (0 – no infection, 1 - infected) at each year for every individual. - Enter ‘Y’ and a random status infection (0 or 1) is created and initialized for each individual. Vertical transmission at given rate below will then occur. - Enter ‘N’ and all individuals will have 0 infection and this module is turned off.
Transmissionprob	0 - 1	This is the transmission probability. If a mother has the infection the chance that the infection will be passed along to the offspring.
3.10 Growth options (see Section 1.3.1.6)		
growth_option	‘N’, ‘known’ ‘vonB’, ‘temperature’, or ‘bioenergetics’	This is the growth function option. Parameters are specified in the following fields. These function can work for either size or age control. If size control, then the size classes are binned following a nearest neighbor routine to the closest size given in the ClassVars.csv file. Then those individuals are assigned a new ‘adjusted’ age.

		- Enter 'N' here to turn off growth and the rest of the growth parameters following are ignored. - Enter 'known' to assign each individuals size by a known amount each year. Given the age of the individual, the size class entered in the ClassVars field is used. This will only be applied at the second DoUpdate. - Enter 'vonB' to use the von Bertalanffy equation for growth. Newsize = size_Loo * (1 – exp(-size_R0 * ('adjusted' age + 1))). - Enter 'temperature' and the von Bertalanffy function is modified by parameters that are fit to temperature as well.
growth_Loo, growth_R0	'266' '0.19'	The von Bertalanffy L infinity and growth rate parameters used to grow individuals via: L = Loo * (1 – exp(– R_0 * (i + 1-t_0))) where age can be the new binned 'adjusted age' if size control is operating.
growth_max, growth_CV, growth_t0		Fit parameters for incremental growth based on temperature. Temperature growth values are specified for two different migration stages (Temperature Out and Temperature Back) in the PatchVars.csv file. See section 1.3.1.6 for equations.

4 Output

Folders will be created in your project directory labeled with a unique time stamp (dos convention), e.g., '..\data\output_test_1332964297\batchrun0mcrun0'. Monte Carlo runs will be designated mcrun0, mcrun1,… with the number of runs specified in the PopVars.csv file. Each batch run will be labeled batchrun0, batchrun1, … with the number of batch runs corresponding to the number of parameter input rows in the PopVars.csv file. The following is a description of the files that can appear in each batchrun{}mcrun{} folder.

4.1 Individual files

4.1.1 ind.csv or indSample.csv

For the specified 'output_years' (or years to output given in the PopVars.csv file), files with information on each individual is reported. For example, if the user specifies a list for the years to report, e.g., '0|4|10', then ind0.csv, ind4.csv, and ind10.csv will be written to each Monte Carlo folder. The first *DoUpdate()* module will automatically write out starting information for each individual to the file ind-1.csv. This is the information for each individual when they are back at their natal grounds.

If the user specifies 'Sample' for the 'gridSampling' option, then indSample0.csv, indSample4.csv, and indSample10.csv will be written to each Monte Carlo folder. This is the information for each individual after they have migrated out (or resided) from their natal grounds. This file will not include the fingerlings (Age0 class) that enters into the population at this point. See Figure 5 flow diagram for 'DoUpdate' processes for order of processes and respective sample of the individuals.

The following is a description of what is included in each of these files:

- 'Subpopulation' – This designates the patch that each individual is sampled in (patch numbers given in the PatchVars.csv file).
- 'XCOORD' and 'YCOORD' – The reference for each patch location given in the PatchVars.csv file.
- 'ID' – a unique identifier for each individual that can be tracked through time. For the ind.csv files: {movement alpha character (see below) and immigrated patch }_{emigrated from patch}_('P' and natal patch born in}}_{'Y' and year

born}_{unique individual number}. E.g., R2_2_P2_Y10_150 would mean that this is the 150th unique individual that was a residor in patch 2 born in year 10 from natal patch 2. Initialization of individuals will have Y-1. Therefore, a query for year, patch, and number will track individual locations. For the indSample.csv files, the {emigrated from patch} is not included. The first alpha character tracks the individuals movement:

  - R – Residor; stayed in current patch,
  - E – Emigrator; moved out from natal patch (note that an individual could attempt to move out, but still settle in the same natal patch),
  - S – Strayer; strayed to another natal ground location (note that an individual could attempt to stray, but settle still in natal patch),
  - I – Immigrator; migrated out and returned back to original natal patch,
  - Z – Attempted to immigrate back to natal grounds, but was forced to stray elsewhere due to either K = 0 at original patch (e.g., catastrophic event) or the movement cost exceeded the specified movement threshold (e.g., the individual traveled too far away from natal grounds).
  - Age0 – emerged offspring within natal patch. This identification will only occur in the ind().csv files.

- ‘sex’ – 0 for female and 1 for male.
- ‘age’ – Age of each individual.
- ‘size’ – Size of each individual.
- ‘mature’ – Whether or not that individual is a reproducing individual.
- ‘capture’ – 0 for not sampled and 1 for sampled.
- ‘recapture’ – 0, 1, 2, … value for how many times this individual was captured.
- ‘infection’ – Whether or not that individual was infected through vertical transmission (see transmission probabilities in Table of input variables for PopVars.csv).
- ‘CDist’ – If this individual was a migrator (Emigrator or Immigrator), then the total cost distance that this individual traveled.
- ‘L0A0, L0A1, …, L0AN, L1A0, L1A1, …, LMAN’ – The diploid genotypes of this individual for N alleles and M loci given. For each locus values of 0, 1, or 2 can be given. A value of 2 represents a homozygous locus at the given allele location, else two 1’s will specify the allele locations for a

heterozygous locus. 0’s will be given for every other allele not present.

4.1.2 Genetics format option

The user also has the option to output genotypes in a number of different formats. See ‘gridformat’ option in the Table of input parameters for these options. For example, if ‘structure’ is specified then the files structure_ind0.csv, structure_ind4.csv, and structure_ind10.csv will be produced.

4.1.3 Genetic distance files

The user can also specify a genetic distance matrix option for every individual and will see GDmatrix0.csv, GDmatrix4.csv, GDmatrix10.csv. Warning: these calculations can take a very long time and recommend external code or packages to calculate these after simulations are complete.

4.2 Summary measures

The following is a description of various output files to summarize each Monte Carlo run either by class, patch and through time.

4.2.1 Time versus class

Each folder will contain a summary file for each year and for various measures across the classes (age or size) labeled ‘summary_classAllTime_{foldertime}.csv’. Every measure through time will be separated by ‘|’ distinguishing the class breaks in ascending order given in the ClassVars.csv file. The following is a description of each measure and in order of processes (see Figure 5):

- Year – This is the time unit (year or generation).
- Ages – A column of age classes.
- N_Initial_Age – N counts for age classes at start of each time unit.
- Age Sizes – Size mean and standard deviation of age classes.
- Size_Classes – A column of size classes.
- N_Initial_Class - N counts for size classes at start of each time unit.
- Class Sizes – Size mean and standard deviation of size classes. **** Note: If size class switch is specified, then the following tracking numbers are then binned by Size_Class above, else binned by Age. ***

- Births – The number of births.
- N_GrowthBack – N counts after the first growth stage – second DoUpdate().
- PackingDeaths_Emigration – The number of individuals that died during the density dependent packing event after migration out of natal grounds (i.e., after DoEmigration()).
- N_Emigration – N counts after DoEmigration().
- Deaths_AfterEmiMort – The number of deaths from the DoMortality() module (constant patch or class deaths) after the DoEmigration() process.
- N_GrowthOut – N counts after the second growth stage – third DoUpdate().
- PackingDeaths_Immigration – The number of individuals that died during the density dependent packing event after migration back to natal grounds (i.e., after DoImmigration()).
- N_Immmigration – N counts after DoImmigration() (age 0s counted here).
- Deaths_AfterImmiMort – The number of deaths from the DoMortality() module (constant patch or class deaths) after the DoImmigration() process.

4.2.2 Class by given time

Using the 'summaryOutput' option, each given year specified will produce a file summarizing measures for each class. They will be laveled 'summary_class{year}_{foldertime}.csv'. All the above section 4.2.1 measures will be listed within these files, separated by class.

4.2.3 Time versus patch

Each folder will contain a summary file for each year and for various measures across the patches (i.e., subpopulations) labeled summary_popAllTime_{foldertime}.csv. Every measure through time will be separated by '|' distinguishing the patch breaks in ascending order given in the PatchVars.csv file. The first number, unless specified otherwise, is the total population value. The following is a description of each:

- Year – This is the time unit (year or generation).
- K - The total carrying capacity.
- N_Initial – N at the beginning of each time unit.
- Size – Patch mean and standard deviation.
- N_Females – The total number of females. First number is total.
- N_Males – The total number of males. First number is total.

- N_BreedFemales – The total number of breeding class (size or age) females.
- N_BreedMales – The total number of breeding class (size or age) males.
- MatureCount – The total number of reproducing age individuals.
- ImmatureCount – The total number of immature individuals.
- EggLayEvents – The number of breeding events for females (can be less than the number of mature females when egg frequency is less than 1).
- Births – The number of eggs or litter size.
- EggDeaths – The number of deaths to eggs or litter. Births – EggDeaths is the survival from egg to fingerling, e.g.
- SelectionDeaths_Emigration – The number of deaths due to differential mortality (spatial selection) during the migration out process.
- MoveDeaths_Emigration – The number of individuals that could not migrate due to high cost to moving to open patches.
- PackingDeaths_Emigration – The number of individuals that died during the density dependent packing event after the migration out process.
- N_Emigration – N counts after DoEmigration() process.
- Deaths_EmiMort – The number of deaths from the DoMortality() module (constant patch or class deaths) after the DoEmigration() process.
- N_EmiMortality – N counts after the DoMortality() module after the DoEmigration() process.
- SelectionDeaths_Immigration – The number of deaths due to differential mortality (spatial selection) during the migration back process.
- MoveDeaths_Immigration – The number of individuals that could not migrate back due to high cost to moving to open patches
- PackingDeaths_Immigration – The number of individuals that died during the density dependent packing event after the migration back process.
- N_Immigration – N counts after the DoImmigration() process or when individuals are settled back into their natal grounds. Age 0s counted here.
- N_ImmiMortality – N counts after the DoMortaltiy() module after the DoImmigration() process.
- Alleles – The number of unique alleles.
- He - The expected heterozygosity value.
- Ho – The observed heterozygosity value.

- p1 – The allele frequency of A or the first allele in the first locus (used for CDEVOLVE).
- p2 – The allele frequency of a or the second allele in the first locus (used for CDEVOLVE).
- q1 – The allele frequency of B or the first allele in the second locus (used for CDEVOLVE).
- q2 – The allele frequency of b or the second allele in the second locus (used for CDEVOLVE).
- MateDist and MateDiststd – The mean and std {cost} distance individuals travel to mate. Note movement option '4', '6' will always be 0 and '9' will be probability values.
- EmigrationDist and EmigrationDist_SD – The mean and std {cost} distance individuals disperse from their natal location separated into female and male movement during the DoEmigration() process. Note movement option '4', '6', and '9' will be probability values. (This is the distance from 'NatalPop' to 'ImmiPop'.)
- FromHomeDist and FromHomeDist_SD – The mean and std {cost} distance individuals moved during the DoImmigration process. This is for all individuals, except 'Age0s', including strayers and immigrators back to natal grounds. This is separated into female and male movement. Note movement option '4', '6' will always be 0 and '9' will be probability values. (This is the distance from 'NatalPop' to 'ImmiPop'.)
- StrayerDist and StrayerDist_SD – The mean and std (cost) distance individuals stray separated into female and male movement during the DoImmigration() process. Note movement option '4', '6' will always be 0 and '9' will be probability values. (This is the distance from 'EmiPop' to 'ImmiPop'.)
- HomeAttemptStrayerDist and HomeAttemptStrayerDist_SD – The mean and std (cost) distance individuals stray after they attempt to return to their natal grounds separated into female and male movement during the DoImmigration() process. Note movement option '4', '6' will always be 0 and '9' will be probability values. (This is the distance from 'EmiPop' to 'ImmiPop'.)
- Infected – The number of individuals that are infected (used for CDINFECT) with given transmission probability (vertical infection).
- Residors – Number of individuals after immigration process that are resident to each natal population.
- Strayers_1 – Number of individuals that strayed into each population after immigration process.

- Strayers_2 – Number of individuals that attempted to immigrate home, but could not, then strayed into each population after immigration process.
- Immigrators – Number of individuals that immigrated.

4.2.4 Patch by given time

Using the 'summaryOutput' option, each given year specified will produce a file summarizing measures for each patch. They will be labeled 'summary_pop{year}_{foldertime}.csv. All of the above section 4.2.3 measures will be listed here, separated by patch.

5 Special Cases

This section provides examples of specific and commonly used simulation scenarios. Parameter values and input setup are given for each.

5.1 Non-overlapping generations

Control non-overlapping generations through the ClassVars.csv file. The ClassVars.csv file must contain two rows for Age 0 (i.e., offspring population) and Age 1 (i.e., mature adult population) classes. Initialize the distribution of Age 0 = 0 and then the distribution of Age 1 = 1. For mortality, use 100% for the 'Mortality Out' option for Age 1. Set the maturation rate at 1 for Age 0 for both female and male. To control a constant population, you must balance your Leslie matrix or to control for a constant population at K, see Wright-Fisher assumptions below.

5.2 Wright-Fisher assumptions

The following is a bullet list of parameters that must be set for Wright-Fisher assumptions.

In the ClassVars.csv file:

- Use two age classes: Age 0 and Age 1.
- Set the distribution for Age 0 to 1 and Age 1 to 0.
- Set the 'mortality out' for Age 0 to 0% and Age 1 to 100%.
- Set all other mortalities to 0%.
- Set the maturation rate for Age 0 to 1 and Age 1 to 0 for both female and male.
- Set the fecundity for Age 0 to 0 and Age 1 to 1.
- Set the 'Percent Female' option to WrightFisher – this will ensure that the initial starting population will have equal females and males in each population, as well as ensuring

that the replacement population is also equal females and males in each population.

In the PatchVars.csv file:

- You must consider a panmictic population: all patches can mate with all other patches and locally disperse or stray freely between patches. Alternately, you could consider one big population or just one patch.

In the PopVars.csv file:

- Set the reproduction to male and female mating with replacement (Y/Y).
- Set the 'offno' choice to 3 for constant litter size
- Set the 'equalClutchSize' to 'N'.
- No mutation.
- Set the egg mortality to 0 and specify 'WrightFisher' in Egg_FemalePercent.

Comparisons of simulation output with theoretical equations illustrate the validation, usefulness, and reliability of such programs. Genetic exchange is simulated in 1 populations with 500. Wright-Fisher assumptions were used for 300 generations, 50 Monte Carlo replicates, and genotypes initialized with 30 loci and 30 alleles per locus. We provide basic graphics, including comparison of results to theoretical predictions for loss of heterozygosity over time (Fig. 7).

He and Ho are calculated at each generation as the estimated proportion of individuals that are expected heterozygous based on a Hardy-Weinberg population and the proportion of individuals that are observed heterozygous, respectively. He and Ho results are shown in Figure 7, compared with heterozygosity produced according to the following equation (Crow and Kimura 1970) adjusted for a small sample size,

$$H_t = (1 - (1 / (2*N_e + 1)))^t * H_o(0), \qquad (20)$$

where the effective population size for separate sexes (Hedrick 2011) is defined as

$$N_e = (4*N_M*N_F) / (N_M + N_F), \quad (21)$$

and H_t is the theoretical rate of decay after t generations, $H_o(0)$ is initial heterozygosity, N_M is the male total, and N_F is the female total.

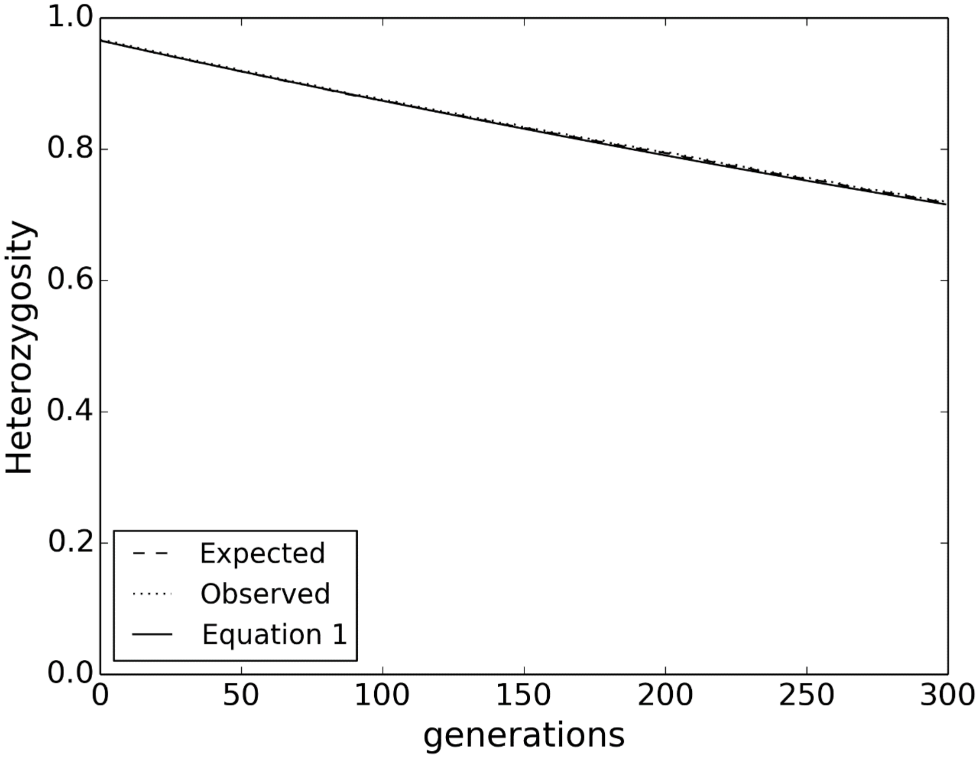


Figure 7: CDMetaPOP heterozygosity. Equation 20 H_t (solid line) show the theoretical rate of decay in the panmictic scenario, as expected for one subpopulation (N = Ne = 500) that matches the simulations observed (dotted-line) and expected (dashed-line) heterozygosity values.

5.3 Deterministic and stochastic variability

Controlling for deterministic versus stochastic variability can be accomplished for most parameters. Here, we show how each parameter can be used for deterministic events with case examples (i.e., sensitivity) of chosen parameters. The following is a table with parameters for certain types of variability.

Table 2: Initial Parameter Space for the WCT model

Parameter	Description	Demographic variability	Spatial variation	Temporal variation	Correlation structures

A_1 ~N(0,1)	Autocorrelation deviate - Random normal deviate used to create autocorrelation structures in other variables	N/A	N/A	Primarily auto-correlation	Specify ϕ as necessary to simulate environmental data (e.g. temperature; ϕ~0.8)
E_1 ~N(0,1)	Environment deviate 1. Used to capture epistemic uncertainty in the model	N/A	N/A	Could be used to simulate trend where μ~f(t)	Specify correlation coefficient rho (ρ) with A_1 as required.
K (μ,σ)	Carrying capacity specified for each patch	N/A	Heterogeneously distributed as a function of patch size and empirical habitat data[1].	Environmental variation simulated as a normal deviate specified with σ.	Since K is primarily a habitat variable, stochasticity is expected to be spatially highly correlated, as well as temporally autocorrelated. Link to E_1
N_t	Initial population size	N/A	Initially distributed as K/2	N/A	N/A
M_{eggs} ~N(μ,σ)	Egg mortality	Random Normal draw based on empirical data	N/A in current model[2]	Implicit in demographic variation	Variation in egg mortality might be expected to be negatively correlated to environmental variation. i.e. when conditions are good mortality should decline.

[1] In later models, spatial distribution of K could be used to simulate other factors such as habitat degradation or competition with other species.

[2] Could be implemented as a function of surrogate variables such temperature or flow

					$\sim f(E_1)$; $\rho \sim -0.5$
$M_{ageclass}$ $\sim N(\mu,\sigma)$	Age/stage specific mortalities	Random Normal draw as specified	Uniform. [3]	Implicit	N/A
M_{out} M_{back} $\sim N(\mu,\sigma)$	Additional mortality associated with movement	Uniform random draw	Explicit patch level control	Implicit as function of σ	$\sim f(E_1)$; $\rho \sim -1$
Migration and Stray rates	Population level controls for migration rate and probability of straying	Uniform random draw	Uniform	Constant	Not implemented but could be explored as a negative correlate of E_1
Growth: L_∞ k t_0 T_optimal	Population variables that are modified as a function of Temperature and duration of growing season	Determin-istic	Growth is individual-ly calculated as a function of Temperature and location	Implicit as a function of environ-mental Temperature	N/A
Maturity	Probability of maturity as a logistic function of length	Uniform random draw	Uniform	Constant[4]	Not implemented but could be explored as environmental ly correlated
Fecundity	Mean number of eggs per female as function of length	Poisson draw	N/A	N/A	N/A

For each input file the parameters can be specified as deterministic as follows.

In the PopVars.csv file:

[3] Spatial structure in age/stage specific mortalities could be used to simulate fishing pressure or other specific source of additional mortality.

[4] Potential target for selective/adaptive control

- Deterministic runs were tested for size control on (‘Y’).
- Movement can be controlled to occur within each patch, or demes/metapopulations with no movement between. Set all movement numbers to ‘6’.
- Deterministic runs were tested for female and male with replacement for mate selection.
- Maturation – in the fields mature_length_female and mature_length_male, choose a length in which individuals will become mature. Then the probability of maturation curves are ignored.
- Fecundity – the egg mean to length relationship can still be used, but set the draw to be a constant mean given size (offno = ‘3’) with equal clutch size (equalClutch == ‘Y’) and egg frequency of 1.
- Egg mortality – set standard deviation to 0.

In the PatchVars.csv file:

- K can be set to any value, but set K StDev to 0.
- N can be set to any value.
- Mortality values – Set all StDev values to 0.
- Movement (migration and straying) is controlled in the PopVars.csv file, so these field will be irrelevant.
- Growth Temperature and grow days – Set all StDev values to 0.

In the ClassVars.csv file:

- Body Size – Enter 0 for the Body Size StDev. This will ensure that individuals are initialized with the given Body Size number.
- Mortalities – Set all mortality StDev values to 0.
- Movement (migration and straying) is controlled in the PopVars.csv file and these fields will be irrelevant.
- Maturation and fecundity – These fields are ignored when size control is operating.

5.3.1 Deterministic example runs

5.3.1.1 Temperature and grow days

Here, we present an example sensitivity run for how stochasticity in temperature and grow days effects results using mean patch growth rate and initial population abundance as responses. 406 patches were used and no movement was considered between patches to simulate 406 independent metapopulations. K was set to 200 for each patch and initial N set at 100. Initial body sizes to set 8 size classes were 31, 83, 130, 162, 181, 192, 199, and 206 mm. Males and females matured at the set given length of 134 mm and 156 mm, respectively. Females had a constant mean egg number (option '3') given their length and following the Downs et al. 1997 exponential equation (Egg_Mean_par1 = 9.55 and Egg_Mean_par2 = 0.018). Egg mortality was set at 62%. Because we were interested in the effect temperature and grow days had on growth, we used the fit age-length-temperature data equation (growth_Loo = 353, growth_R0 = -0.57, growth_temp_max = 13, growth_temp_CV = 0.33, growth_temp_T0 = -0.196). We allowed packing to operate.

We considered 4 scenarios using the PatchVars file information with the temperature values Out = 2.95 and Back = 11.18, and grow day values of Out = 242 and Back = 123 for each patch: (1) Constant temperature (TConst) and constant grow days (GDConst), (2)Variable temperatures (TVar) around means given and constant grow days (GDConst), (3) Constant temperature (TConst) and variable grow days (GDVar) around means given, and (4) Variable temperatures (TVar) and variable grow days (GDVar) around means given above. Each scenario was run for 20 years with 20 Monte Carlo replicates. Figure 8 and 9 show mean patch growth rate and initial population N results for the 20 replicates (confidence intervals too small to be seen at this scale).

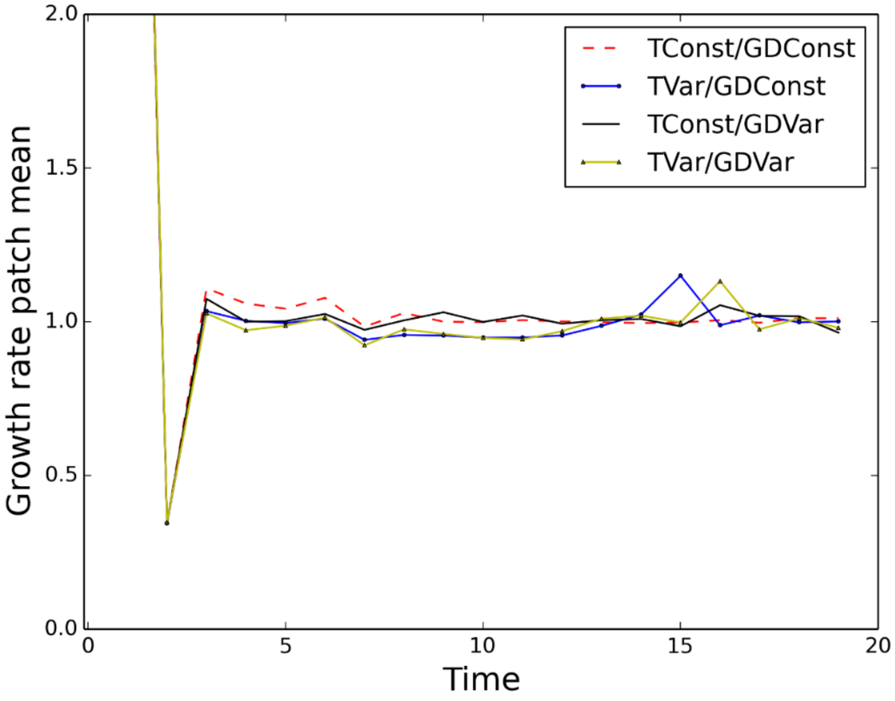


Figure 8: Mean patch growth rate for sensitivity around temperature and grow days at the patch level. TConst – constant temperature, GDConst – constant grow days, TVar – variable temperature, GDVar – variable grow days.

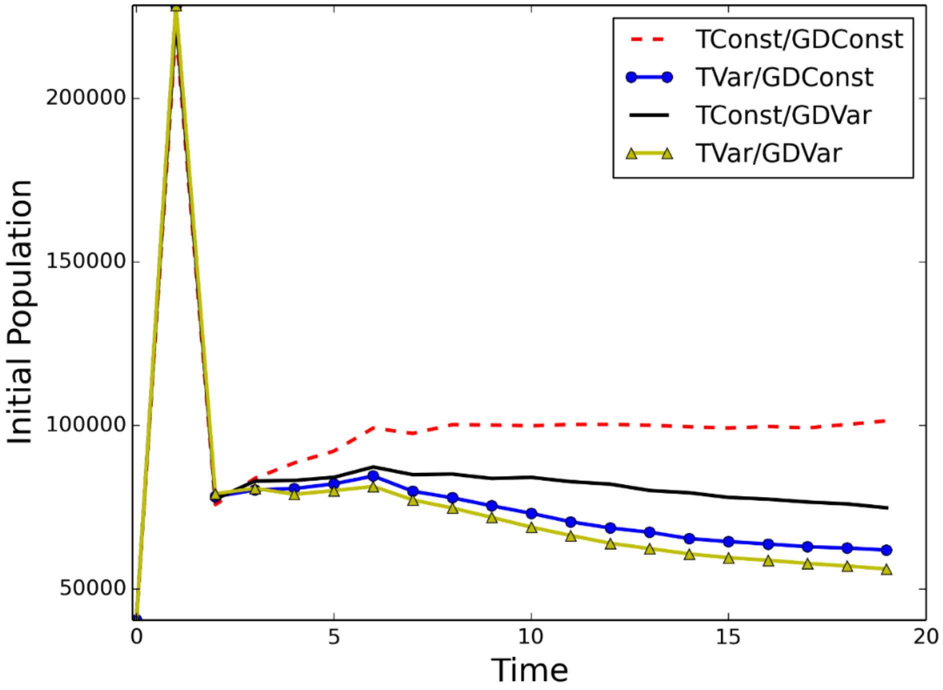


Figure 9: Initial population N for sensitivity around temperature and grow days at the patch level. TConst – constant temperature, GDConst – constant grow days, TVar – variable temperature, GDVar – variable grow days.

5.3.1.2 K / patch

Using all of the same parameters as mentioned in 5.3.1.2, while holding temperature and grow days constant, we considered 5 scenarios for starting K values: K = 100, 300, 500, 700, 900. Each scenario was run for 20 years with 20 Monte Carlo replicates. Figure 10 and 11 show mean patch growth rate and initial population N results for the 20 replicates (confidence intervals too small to be seen at this scale).

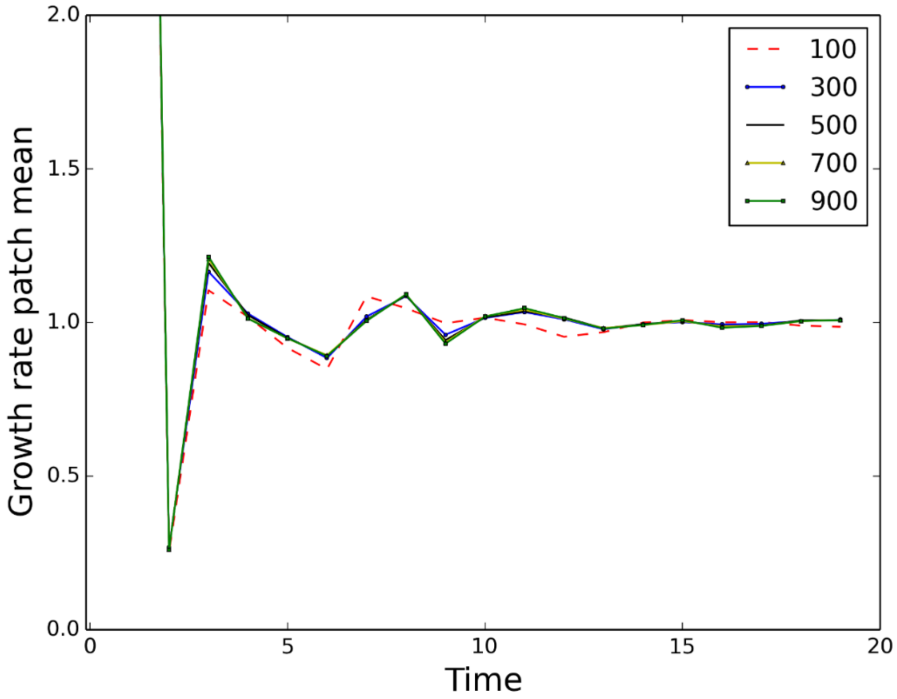


Figure 10: Mean patch growth rate for sensitivity around starting K values.

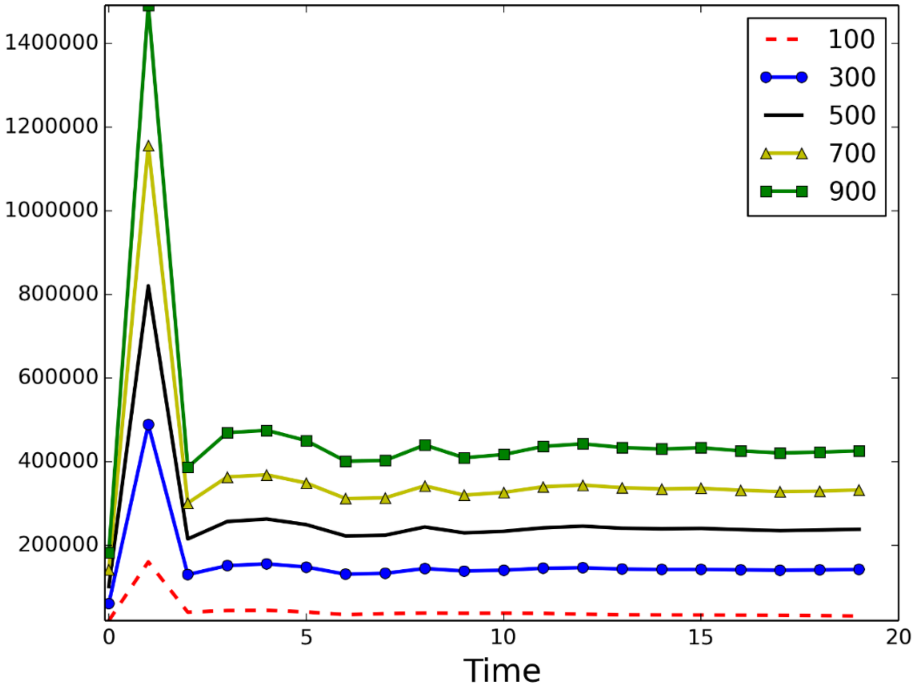


Figure 11: Initial population N for sensitivity around starting K values.

5.3.1.3 Initial N

Using all the same deterministic parameters mentioned above and K = 200, we looked at 5 scenarios for sensitivity around starting N: N0 = 25, 50, 100, 150, and 175. Each scenario was run for 20 years with 20 Monte Carlo replicates. Figure 12 and 13 show mean patch growth rate and initial population N results for the 20 replicates (confidence intervals too small to be seen at this scale).

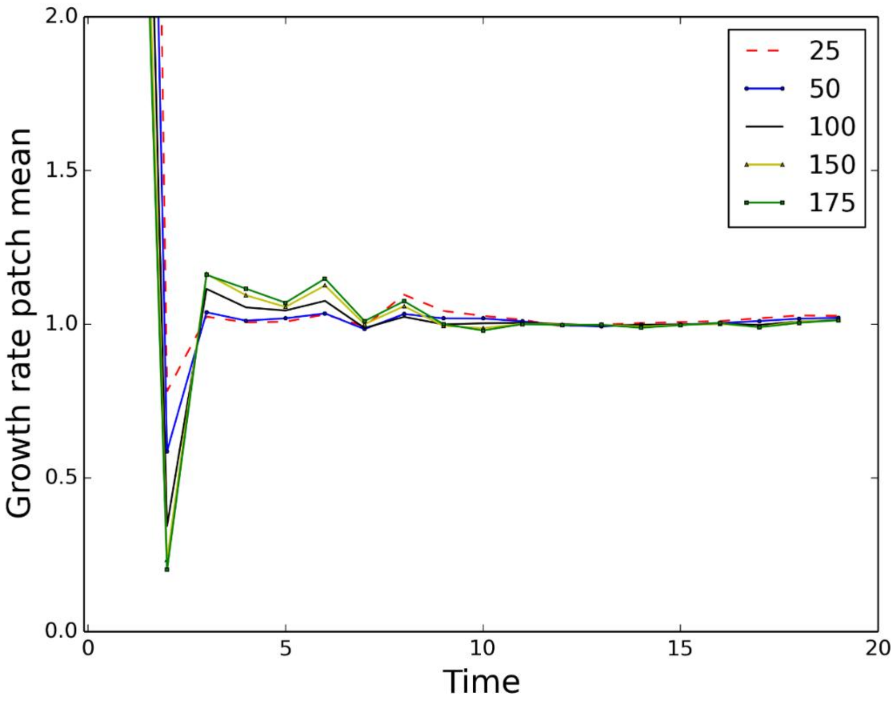


Figure 12: Mean patch growth rate for sensitivity around starting N0 values.

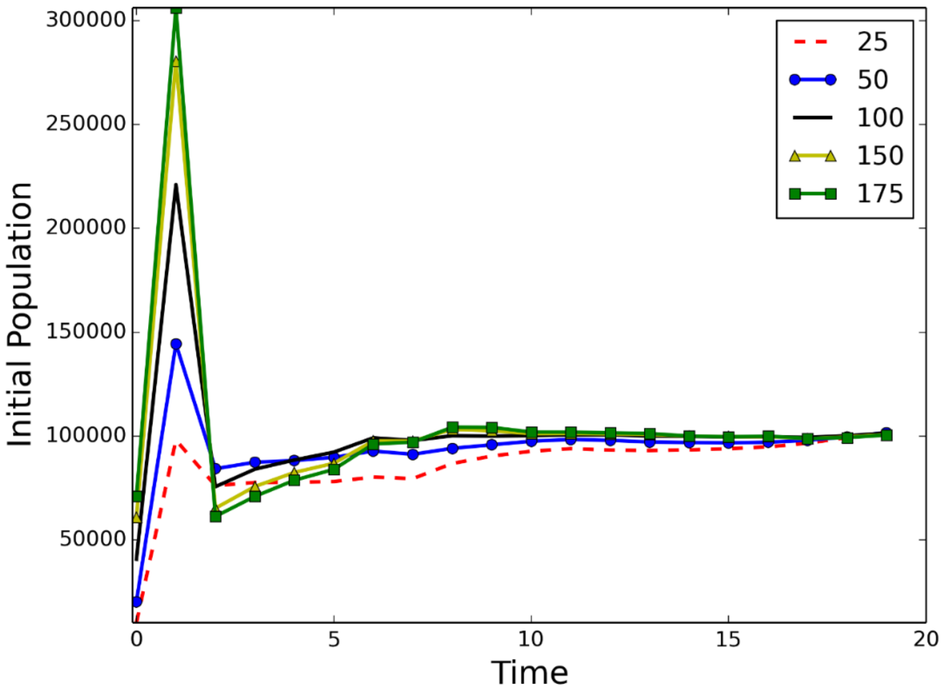


Figure 13: Initial population N for sensitivity around starting N0

5.4 Terrestrial Example

CDMetaPOP was specifically developed for species living in dynamic riverine landscapes ('riverscapes'; Ward 1998; Fausch et al. 2002). However, the model is suitable for a spectrum of spatial processes that operate in any spatially heterogeneous environment (e.g., terrestrial; Landguth et al. 2015 or aquatic; Schlosser 1995; Rieman and Dunham 2000) where spatial arrangement of habitats, habitat-specific demographic variables, and behavioral variables describing the dispersal characteristics of the species can be defined (Pulliam et al. 1992). Each of the modules described in this user manual can be essentially turned on or off depending on the objective of the simulation study. Here, we show a terrestrial example using Whitebark Pine (WBP; *Pinus Albicaulus*) and the added influence of White Pine Blister Rust (WPBR; *Cronartium ribicola*) mortality with resistant genetics in the US Northern Rocky Mountains.

We created a WBP system model that required the development and parameterization of a number of species-specific processes (see Fig. 14). After initialization of the model (e.g., stands, stage structure, and genetics; where stands now refer to 'patches'), pollen dispersal was considered during the summer in which age 0 fertilization event occurs. Then, age 1 cones from the previous year's fertilization event began to emerge on each tree and dispersed in the fall. Overwinter, stage-structured density independent mortality is considered while checking each stand’s carrying capacity. Growth and new mature individuals occurred by spring with the additional WPBR mortality on mature individuals implemented. More detailed methods with data sources for developing the model are outlined below.

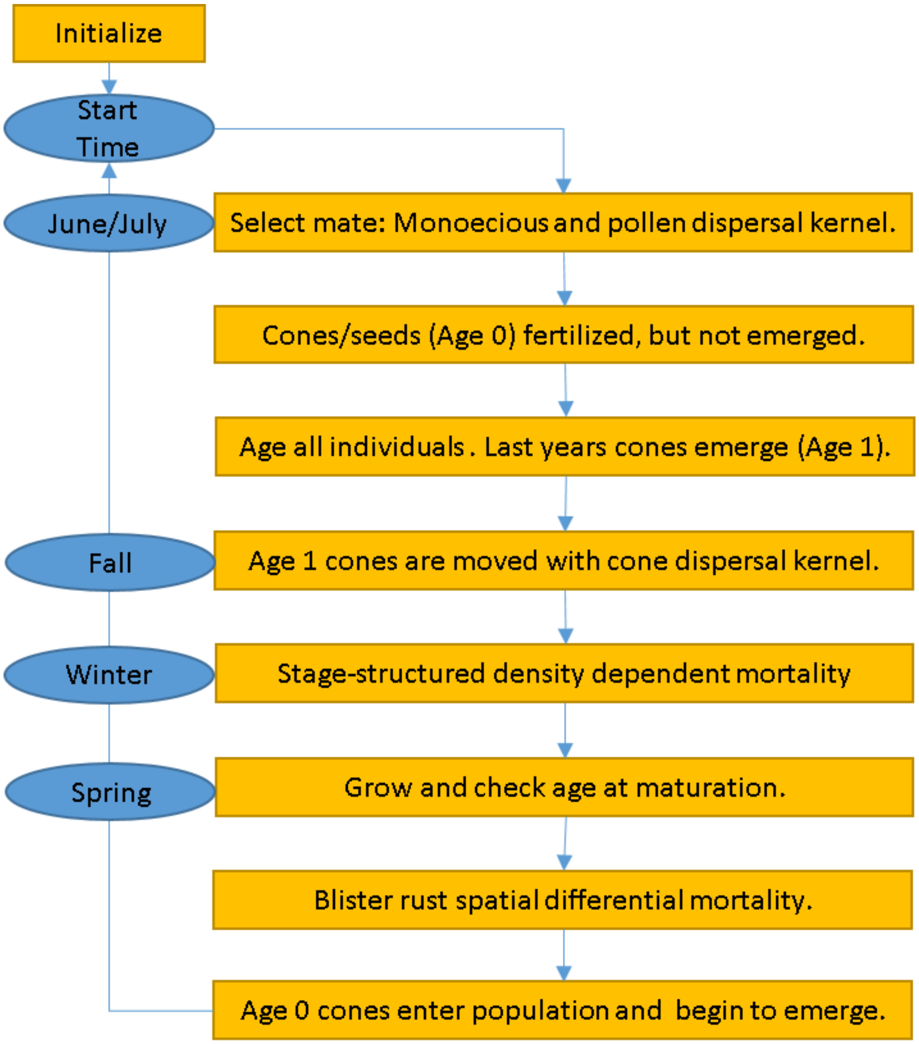

Initialize
Start Time
June/July
Select mate: Monoecious and pollen dispersal kernel.
Cones/seeds (Age 0) fertilized, but not emerged.
Age all individuals . Last years cones emerge (Age 1).
Fall
Age 1 cones are moved with cone dispersal kernel.
Winter
Stage-structured density dependent mortality
Spring
Grow and check age at maturation.
Blister rust spatial differential mortality.
Age 0 cones enter population and begin to emerge.

Figure 14. CDMetaPOP flow diagram with respect to Whitebark Pine major processes.

Stands and carrying capacity: The WBP simulation extent constrained to the Northern Rockies seed zones extent (Mahalonvich and Hopkins 2011; Fig. 15) contained 1,059 initial spatially-delineated stand locations separated by 5 km. Presence or absence of WBP was determined by a 0.5 probability cutoff using habitat suitability data (Landguth et al. 2015). We assumed a carrying capacity (K) of 100 at each stand location.

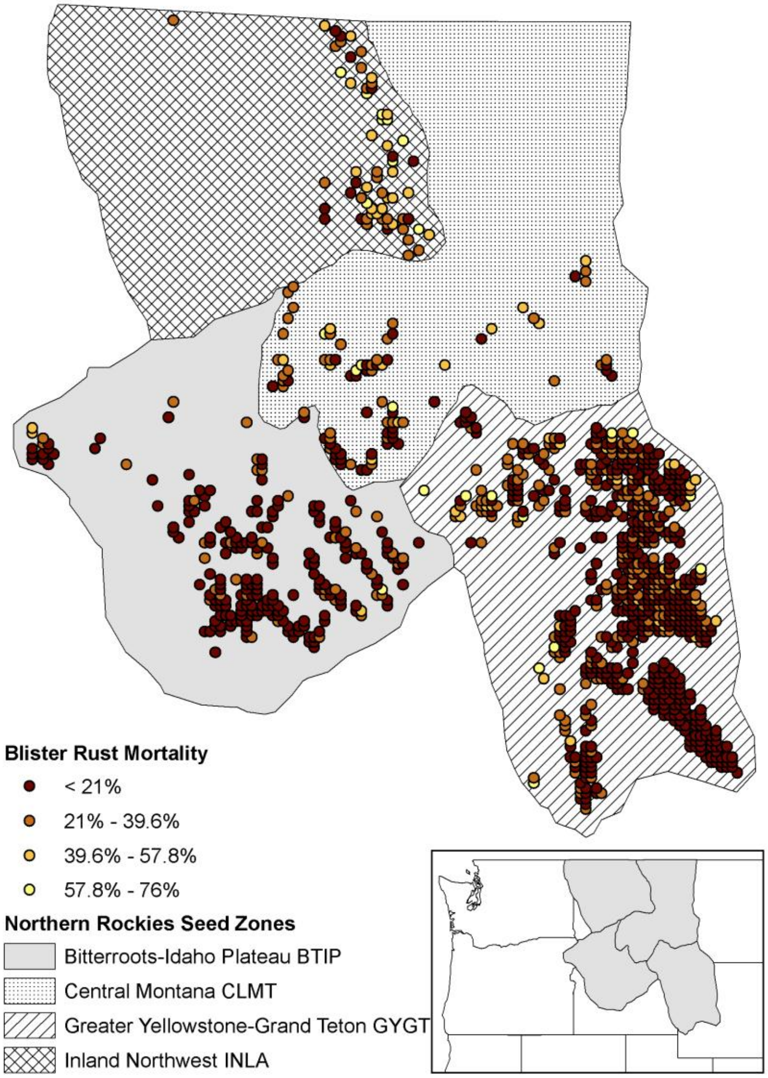


Figure 15: Study area defined by the northern Rockies seed zones (Malahovich and Hopkins 2011) with initial 1,059 stand locations. WPBR relative spatial selection mortality shown for each stand.

Stage classes: We initialized the model at time = 0 with a random distribution of 500 age classes. After 25 years, a stable age distribution was reached at which time we began genetic exchange. We assumed that age 0 individuals were fertilized individuals and 12 months later emerged as age 1 cones on the trees that could then be dispersed. Size classes increased monotonically at a diameter at breast height (DBH) of 0.2 cm each year (Keane et al. 2007). As trees progressed through each size class, size-linked parameters (e.g., probability of mortality, probability of maturation, and fecundity) varied.

Genetics: Genetic exchange was initiated 25 years into the simulations in order to stabilize the population dynamics first before introducing spurious genetic bottlenecks due to transient population dynamics. We initialized the model with allele frequency files generated for each seed zone (Mahalonvich and Hopkins 2011) that comprised of 16 loci with at most 9 polymorphic alleles per locus. In addition, we added a selection-driven locus. This locus was putatively adaptive with 0.01 and 0.99 frequency for the first and second allele, respectively (see below description on White Pine Blister Rust selection for more details on how adaptive genetic processes were implemented). We did not consider mutation, which is reasonable considering the short simulation time period.

Maturation and fecundity: Mature individuals were assumed to be > age 20. Although WBP may typically take longer to reach maturity when growing at poor sites, we used a lower bound of 20 years drawn from literature in order to allow for more generations in the short time frame simulated (Fire Effects Information System, accessed September 2015). A size-based fecundity model was implemented to determine the number of seeds produced at a given basal area per stand following the individual tree DBH conversion to basal area: Basal Area = 0.00007854 * DBH2 (Keane et al. 2007). Then, we used the value of 500 cones per 1 basal area (Barringer et al. 2012) multiplied by 20 seeds per cone, which yielded a size-based seed production per individual tree.

Mortality: Mortality in the model is regulated by both density-dependent and density-independent processes. In order to simplify processes and isolate the effects of White Pine Blister Rust mortality, we only considered density-independent processes using class-based mortalities. We applied a 99% probability of mortality to age 0 class to mimic 1% probability of seed survival (Izlar 2002). We implemented a cumulated 35% probability of survival for age classes 1 – 15 (Izlar 2002). Trees age 500 and older underwent a 25% probability of survival, which allowed for occasional long lived trees given the length of the simulation time. If a stand happened to reach K, then a random removal of excess individuals was conducted (e.g., Balloux 2001).

Reproduction and pollen dispersal: Reproduction within and across stands was monecious with selfing allowed. Once a pollen spore settled at a respective female counterpart, a fertilization event was assumed, meaning gene exchange was considered and stored as age 0 individuals (not contributing to K) while incubation for 12 months would occur. Pollen movement during summer months was assumed to follow an isolation-by-distance model: probability of pollen dispersal to a respective female cone locations was a function of the inverse-square Euclidean distance with a 50 % maximum study area distance threshold (450 km).

Cone/seed dispersal: Emerged age 1 cones from the previous year were dispersed from individual trees following an isolation-by-distance movement pattern similar to pollen dispersal: probability of cone dispersal to a new stand location was a function of the inverse-square Euclidean distance with a 30 km maximum distance threshold (Lorenz et al. 2011). This produced the majority of cones staying in the same stand or nearest neighbor stands (i.e., dropping near parent tree) with occasional longer distance cone dispersal (e.g., Clark’s Nutcracker). We assumed a cone produced 20 seeds. Seed survival was assumed to be 1 % and the ability for a seed to establish in a new stand location was determined based on resource availability (i.e., carrying capacity not exceeded in stand location cone dispersed to).

White Pine Blister Rust mortality and resistance: We used WPBR probability of occurrence map values (Landguth et al. 2015) at each stand as a proxy for differential mortality applied to mature trees only (Fig. 1). Depending on the simulation scenario (see following section), values at each stand were implemented depending on the genotype of each individual. This allowed us to isolate the process for a simple model of WPBR resistance, by defining a priori a locus under selection with a genotype being selected for.

Simulation scenarios and analysis: We conducted simulation scenarios to check the overall influence of WPBR presence via added mortality at each stand and then, introduce a gene that was resistant to this mortality (Table 3). We ran simulations for 130 years, with the first 25 years considered ‘burn-in’ for the population dynamics and to stabilize age distributions. We plotted overall mean population abundance, allelic diversity, and heterozygosity for each scenario and used 10 replicates to assess variation in each metric.

Table 3: Simulation scenarios

	Scenario Name	**Description**
WPBR mortality	No Mortality	The null model in which no WPBR mortality considered.
	All Stands Gene Resistance	All stand locations applied WPBR mortality (Fig. 1). One genotype assumed to be resistant to WPBR.
	All Mortality	All stand locations applied WPBR mortality (Fig. 1) regardless of genetic makeup.

Results: Overall population mean abundance for each scenario is shown in Figure 16. 'No Mortality' (Fig. 16; dashed black line) shows stable population dynamics with a declining population for the extreme scenario 'All Mortality' (Fig. 16 red line). When we introduced a gene for WPBR selection applied at every stand (Fig. 16 blue line), population abundance still declined, but higher than the 'All Mortality' scenario. Overall population mean allelic diversity is shown in Figure 17 and shows similar patterns to that of the population abundance graphs.

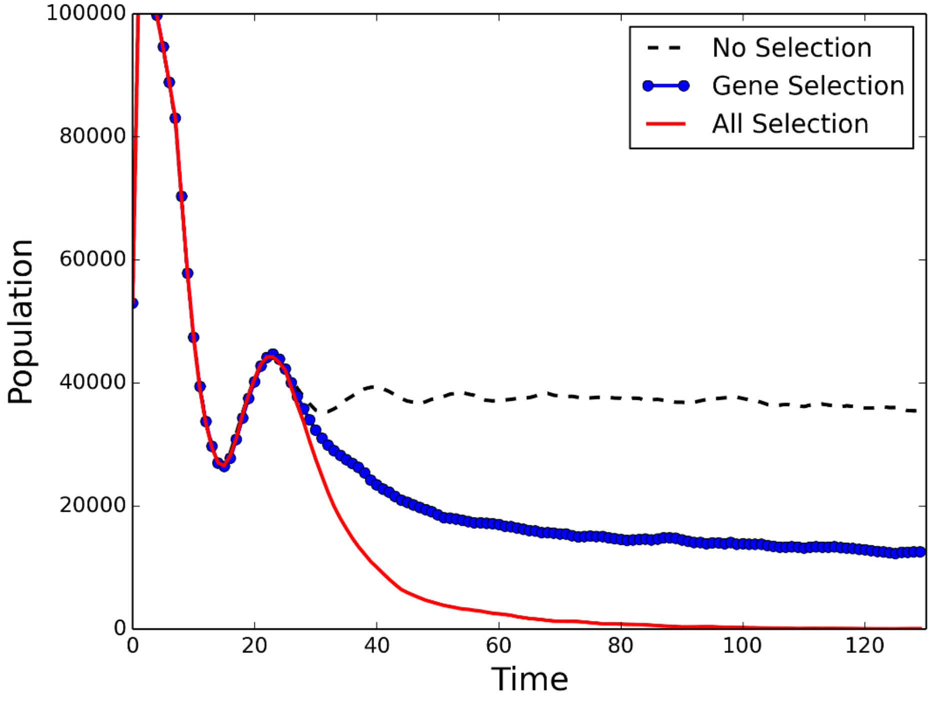


Figure 16. Population abundance through time for each scenario.

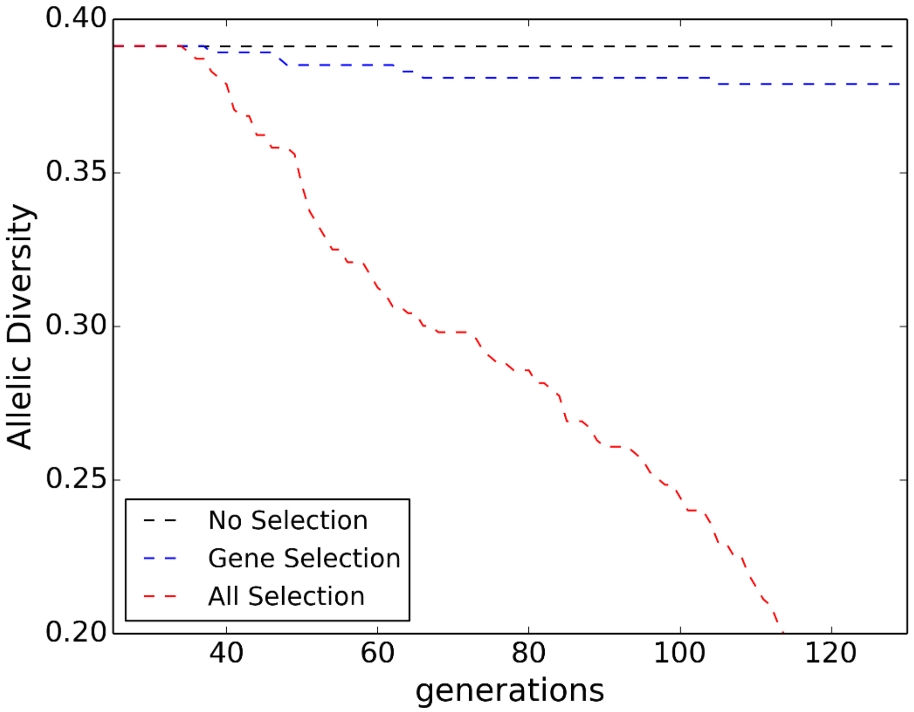


Figure 17. Allelic diversity through time for each scenario.

6 General Issues

6.1 How to obtain CDMetaPOP

The program is freeware and can be downloaded at http://cel.dbs.umt.edu/software/CDMetaPOP/ with information for users, including manual instructions, FAQ, publications, ongoing research, and developer involvement.

6.2 Debugging and troubleshooting

For help with installation problems please check first for postings at our web site. Otherwise, please report problems including any bugs, to me at erin.landguth@mso.umt.edu.

6.3 How to cite CDMetaPOP

The reference to cite is as follows, substituting the version number:

Landguth EL, Bearlin A, Day C, Dunham J (XXXX) CDMetaPOP: an eco-evolutionary metapopulation simulation model for population viability analysis in landscape and riverscape genetics. v1.0. X, X, X-X.

6.4 Disclaimer

The software is in the public domain, and the recipient may not assert any proprietary rights thereto nor represent it to anyone as other than a University of Montana-produced program (version 1.x). CDMetaPOP is provided "as is" without warranty of any kind, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The user assumes all responsibility for the accuracy and suitability of this program for a specific application. In no event will the authors or the University be liable for any damages, including lost profits, lost savings, or other incidental or consequential damages arising from the use of or the inability to use this program.

We strongly urge you to read the entire documentation before ever running CDMetaPOP. We wish to remind users that we are not in the commercial software marketing business. We are scientists who recognized the need for a tool like CDMetaPOP to assist us in our research on landscape ecology issues. Therefore, we do not wish to spend a great deal of time consulting on trivial matters concerning the use of CDMetaPOP. However, we do recognize an obligation to provide some level of information support. Of course, we welcome and encourage your criticisms and suggestions about the program at all times. We will welcome questions about how to run CDMetaPOP or interpret the output only after you have read the entire documentation. This is only fair and will eliminate many trivial questions. Finally, we are always interested in learning about how others have applied CDMetaPOP in ecological investigation and management application. Therefore, we encourage you to contact us and describe your application after using CDMetaPOP.

We hope that CDMetaPOP is of great assistance in your work and we look forward to hearing about your applications.

7 References

Balloux F. (2001) EASYPOP (version 1.7): a computer program for population genetic simulations. Journal of Heredity, 92, 301–302.

Barringer, L.E., Tomback, D.F., Wunder, M.B., McKinney, S.T. (2012) Whitebark Pine Stand Condition, Tree Abundance, and Cone Production as Predictors of Visitation by Clark's Nutcracker. PLOS ONE. doi: 10.1371/journal.pone.0037663.

Bear EA (2005) Effects of temperature on survival and growth of westslope cutthroat trout and rainbow trout: Implications for conservation and restoration. Montana State University Master of Science Thesis.

Bowcock AM, Ruiz-Linares A, Tomfohrde J et al. (1994) High resolution of human evolutionary trees with polymorphic microsatellites. Nature, 368, 455–457.

Bowerman T, "A Multi-Scale Investigation of Factors Limiting Bull Trout Viability" (2013). All Graduate Theses and Dissertations. Paper 1524. http://digitalcommons.usu.edu/etd/1524

Bertalanffy, L. von (1938) A quantitative theory of organic growth(Inquiries on growth laws. II). Human Biol. 10, 181-213.

Castillo JA, Epps CW, Davis AR, Cushman SA (2014) Landscape effects on gene flow for climate-sensitive montane species, the American pika. Molecular Ecology, 23, 843-856.

Downs CC, White RG, Shepard BB (1997) Age at sexual maturity, sex raio, fecundity, and longevity of isolated headwater populations of westlope cutthroat trout. North American Journal of Fisheries Management, 17, 85-92.

Dyer R (2014) popgraph: This is an R package that constructs and manipulates population graphs. http://cran.r-project.org/web/packages/popgraph/index.html

Ettan J van (2015) gdistance: Distances and routes on geographical grids. http://cran.r-project.org/web/packages/gdistance/index.html.

Elith J., Leathwick J.R. (2009). "Species distribution models: ecological explanation and prediction across space and time". Annual Review of Ecology, Evolution, and Systematics40: 677–697. doi:10.1146/annurev.ecolsys.110308.120159

Frank, B.M., Piccolo, J.J., & Baret, P.V. (2011) A review of ecological models for brown trout: towards a new demogenetic model. Ecology of Freshwater Fish, 20, 167-198.

Holden ZA, Abatzoglou JT, Luce CH, Baggett LS (2011) Empirical downscaling of daily minimum air temperature at very fine resolutions in complex terrain. Agricultural and Forest Meterology 151, 1066-1073.

Izlar, D.K. (2002) Assessment of Whitebark Pine seedling survival for Rocky Mountain plantings. M.S. Thesis Humboldt State Univeristy.

Jones MR, Forester BR, Teufel AI, Adams RV, Anstett DN, Goodrich BA, Joost S, Manel S, Landguth EL (2014) Integrating spatially-explicit approaches to detect adaptive loci in a landscape genomics context. Evolution (Special Issue: Evolutionary Landscape Genetics), 67, 3455-3468.

Keane, R., K. Gray and L. Dickinson (2007). Whitebark Pine Diameter Growth Response to Removal of Competition. RMRS General Technical Report Research note RN-32.

Labonne, J., Ravigné, V. Parisi, B., & Gaucherel, C. (2008) Linking dendritic network structures to population demogenetics: the downside of connectivity. Oikos, 117, 1479-1490.

Landguth EL, Cushman SA (2010) CDPOP: a spatially explicit cost distance population genetics program. Molecular Ecology Resources, 10, 156-161.

Landguth EL, Cushman SA, Schwartz MK, Murphy M, McKelvey KS, Luikart G (2010) Quantifying the lag time to detect barriers in landscape genetics. Molecular Ecology, 19, 4179-4191.

Landguth EL, Balkenhol N (2012) Relative sensitivity of neutral versus adaptive genetic data for assessing population differentiation. Conservation Genetics, 13, 1421-1426.

Landguth EL, Muhlfeld CC, Luikart G (2012a) CDFISH: an individual-based, spatially-explicit, landscape genetics simulator for aquatic species in complex riverscapes. Conservation Genetics Resources, 4, 133-136.

Landguth EL, Cushman SA, Johnson NA (2012b) Simulating natural selection in landscape genetics. Molecular Ecology Resources, 12, 363-368.

Landguth EL, Hand BK, Glassy JM, Cushman SA, Sawaya M (2012c) UNICOR: a species corridor and connectivity network simulator. Ecography, 12, 9-14.

Landguth EL, Fedy B, Garey A, Mumma M, Emel S, Oyler-McCance S, Cushman SA, Wagner HH, Fortin MJ (2012) Effects of sample size, number of markers, and allelic richness on the detection of spatial genetic pattern. Molecular Ecology Resources, 12, 276-284.

Lorenz, T.J.; Sullivan, K.A.; Bakian, A.V.; Aubry, C.A. 2011. Cache-site selection in Clark’s nutcracker (*Nucifraga columbiana*). Auk. 128, 237–247.

Mahalovich, M.F., Hipkins, V.D. (2011) Molecular genetic variation in whitebark pine (*Pinus albicaulis Engelm.)* in the Inland West. In: Keane, R.E. (ed.) "High-Five Symposium: The Future of High-Elevation Five-Needle White Pines in Western North America. 2010 June 28-30, Missoula, MT, USA. Proc: RMRS-P-63. For Collins, CO: USDA Forest Service, Rocky Mountain Research Station, p. 124-139.

McRae BH (2006) Isolation by resistance. Evolution 60:1551-1561.

Oliphant TE (2006). Guide to NumPy, Provo, UT., Brigham Young University, http://www.tramy.us/.

Oyler-McCance SJ, Fedy BC, Landguth EL (2013) Sample design effects in landscape genetics. Conservation Genetics, 14, 275–285.

Peacock MM, Dochtermann NA (2012) Evolutionary potential but not extinction risk of Lahontan cutthroat trout 9Oncorhynchus clarkia henshawi) is associated with stream characteristics. Can. J. Fish. Aquat. Sci. 69, 615-626.

Peakall R and Smouse PE (2006) GENALEX 6: genetic analysis in Excel. Population genetic software for teaching and research. Molecular Ecology Notes. 6, 288-295.

Peakall R and Smouse PE (2012) GenAlEx 6.5: genetic analysis in Excel. Population genetic software for teaching and research-an update. Bioinformatics 28, 2537-2539.

Piper et al. 1982

Pritchard JK, Stephens M, Donnelly PJ (2000) Inference of population structure using multilocus genotype data. Genetics 155: 945-959.

Prunier J.G., Kaufmann B., Fenet S., Picard D., Pompanon F., Joly P., Lena J.P., 2013. Optimizing the trade-off between spatial and genetic sampling efforts in patchy populations: towards a better assessment of functional connectivity using an individual-based sampling scheme. Molecular Ecology 22, 5516–5530

R Development Core Team (2012) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.

Ray N (2005) PATHMATRIX: a GIS tool to compute effective distances among samples. Molecular Ecology Notes 5: 177-180.

Ray C, Peacock MM, Dunham JB (2007) (2007). Demography and population dynamics of Lahontan cutthroat trout (Oncorhynchus clarkii henshawi) stream populations in eastern Nevada. Reno, United States Fish and Wildlife Service. Cooperative Agreement FSW 14-48-0001-95646.

Raymond M. & Rousset F, 1995. GENEPOP (version 1.2): population genetics software for exact tests and ecumenicism. J. Heredity, 86:248-249

Rousset, F., 2008. Genepop'007: a complete reimplementation of the Genepop software for Windows and Linux. Mol. Ecol. Resources 8: 103-106.

Rideout RM, Tomkiewicz J (2011) Skipped spawning in fishes: More common than you might think. Marine and coastal fisheries: Dynamics, management, and ecosystem science, 3:1, 176-189.

Rideout and Tomkiewicz 2014

Rossum, G van. and Drake, F. L. (eds), 2006. Python Reference Manual, Python Software Foundation,. http://docs.python.org/ref/ref.html.

Schill DJ, LaBar GW, Mamer ERJM, Meyer KA (2010) North American Journal of Fisheries Management 30:1352–1363.

Scipy: Open Source Scientific Tools for Python [computer program] 2001.

Todd CR, Ng MP (2001) Generating unbiased correlated random survival rates for stochastic population models. Ecological Modelling 144, 1-11.

West Fork Environmental. 2013. Salmonid tissue sampling in the Boundary Hydroelectric Project Area, NE Washington. Prepared for: City of Seattle, Seattle City Light Department

Wright S (1932) The roles of mutation, inbreeding, crossbreeding and selection in evolution. Proceedings of the Sixth International Congress of Genetics, 1, 356–366.

8 Acknowledgements

We thank Helen Neville, Dan Dauwalter, Seth Wegner, Harry Rich, Al Solonsky, and Robin Bjork for fruitful discussions during model development. We thank the class members of FNR 691 from Purdue University, and three anonymous reviewers for helpful manuscript comments. We thank Zack Holden for data contributions for example applications. We thank Lucas Nathan, Marco Alejandro Escalante, and Patrick James for beta testing. Funding for this research comes from Seattle City Light and NASA grant NNX14AC91G.